UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFROMATIKO

Robert Likar

Vizualizacija socialnih omrežij v okviru obogatene spletne aplikacije

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: izr. prof. dr. Marko Bajec

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00448/2009
Datum: 05.04.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ROBERT LIKAR**

Naslov: **VIZUALIZACIJA SOCIALNIH OMREŽIJ V OKVIRU OBOGATENE SPLETNE APLIKACIJE**

VISUALIZATION OF SOCIAL NETWORKS WITHIN A RICH INTERNET APPLICATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

V diplomskem delu obdelajte problematiko pri izbiri primernega orodja za razvoj obogatenih spletnih aplikacij. Izhajajte iz pregleda teoretičnih osnov posameznih vodilnih tehnologij, ki so trenutno dostopne na trgu programske opreme. Izbiro utemeljite ter opišite razvoj in delo z izbrano tehnologijo. Opredelite pojem socialnega omrežja ter opišite izbiro in integracijo komponente za vizualizacijo socialnih omrežij v sklopu obogatene spletne aplikacije.

Mentor:

prof. dr. Marko Bajec

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

IZJAVA O AVTORSTVU
diplomskega dela

Spodaj podpisani Robert Likar

z vpisno številko 63980252

sem avtor diplomskega dela z naslovom:

Vizualizacija socialnih omrežij v okviru obogatene spletne aplikacije

S svojim podpisom zagotavljam, da :

- sem diplomsko delo izdelal samostojno pod mentorstvom

  izr. prof. dr. Marko Bajec

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki ''Dela FRI''.

V Ljubljani, dne 29.09.2009 Podpis avtorja:

Zahvala

Zahvaljujem se svojemu mentorju za koordinacijo ter strokovno pomoč pri izdelavi te diplomske naloge.

Hvala tudi vsem, ki so mi pri tej diplomski nalogi kakor koli pomagali in svetovali, še posebej gre zahvala Maticu, za vso nesebično pomoč in nasvete. Hvala tudi vsem ostalim sodelavcem v podjetju Optilab d.o.o.

In hvala tebi, Alenka, ker si mi vedno stala ob strani in me vseskozi podpirala. V vseh mojih vzponih in padcih verjameš vame in me pri vsem optimistično spodbujaš.

Kazalo

Seznam uporabljenih kratic in simbolov

AJAX	Asynchronous JavaScript and XML - Asinhroni JavaScript in XML
AWT	Abstract Window Toolkit – Java orodje za izdelavo oken
CSS	Cascading Style Sheet – stilne predloge za oblikovanje spletnih strani
DHTML	Dynamic HyperText Markup Language – dinamični označevalni jezik za oblikovanje spletnih dokumentov
DOM	Document Object Model – objektni model sestave dokumenta spletne strani
FDS	Flex Data Services – Flex podatkovne storitve
GWT	Google Web Toolkit – Google-ovo ogrodje za izdelavo AJAX aplikacij
HTML	HyperText Markup Language – označevalni jezik za oblikovanje spletnih dokumentov
JDK	Java Development Kit – java razvojni komplet
JIT	Just in Time – sprotno
JRE	Java Runtime Environment – javansko zaganjalno okolje
JVM	Java Virtual Machine – javanski navidezni stroj
LAMP	Linux, Apache, MySQL, PHP/Perl/Python – Skupek programov za izdelavo dinamičnih spletnih strani
MXML	Macromedia eXtensible Markup Language – Macromediin razširljivi označevalni jezik
RIA	Rich Internet Application – obogatena spletna aplikacija
SDK	Software Development Kit – programski razvojni komplet
SWF	Shockwave Flash File – format za multimedijsko vektorsko grafiko
URL	Uniform Resource Locator – spletni naslov
WPF	Windows Presentation Foundation – Microsoft-ov grafični uporabniški vmesnik za razvoj RIA ter namiznih aplikacij
XAML	Extensible Application Markup Language – razširljivi aplikacijski označevalni jezik
API	Application Programming Interface – aplikacijski programski vmesnik
XML	eXtensible Markup Language – razširljivi označevalni jezik

Povzetek

S porastom popularnosti svetovnega spleta, hitrejšimi povezavami in zmogljivejšimi brskalniki se razvija tudi trg spletnih aplikacij, ki postajajo vse bolj raznovrstne in izpopolnjene. Obogatene spletne aplikacije z izpopolnjeno in vse boljšo funkcionalnostjo počasi izpodrivajo namizne aplikacije. Namen te diplomske naloge je izbira primerne tehnologije za izdelavo obogatenih spletnih aplikacij ter izbira primerne komponente za vizualizacijo socialnega omrežja. V tej nalogi je, poleg različnih naštetih tehnologij za razvoj RIA, v ta namen podrobneje opredeljena uporaba orodja GWT. Orodje GWT omogoča enostavno izdelavo spletnih aplikacij v samo enem jeziku, v Javi. Je zelo primerno za razvoj poslovnih spletnih aplikacij, predvsem zaradi prednosti združevanja programske kode na strežniškem delu ter odjemalčevi strani v skupen projekt.

Uporaba socialnih mrež je idealen način za učinkovit in intuitiven pregled ljudi in povezav med njimi. Vizualizacija socialnih mrež pa igra pomembno vlogo pri analizi in raziskovanju le-teh. Je več kot le skopo prikazovanje zanimivih podob, tu gre predvsem za generiranje učljivih situacij ter odkrivanje pomembnih informacij. V ta namen je bila izbrana komponenta Birdeye RaVis, izdelana s tehnologijo Adobe Flex. RaVis je močna in zelo domišljena komponenta za izris grafov ali omrežij, ki kot glavni podatkovni vir uporablja XML datoteke. Z uporabo vmesnika GWT-Fabridge pa smo poskrbeli, da sama komunikacija med dvema različnima tehnologijama, kot sta GWT in Flex ni predstavljala prevelikih težav.

Ključne besede: RIA, GWT, socialne mreže, RaVis, GWT-FABridge

Abstract

Together with the growth of the World Wide Web's popularity, connections become faster, browsers more efficient, and the market of web applications develops, increasing the web applications' versatility and their level of perfection. Rich web application offer better functionally and are slowly taking the place of desktop applications. The aim of this diploma thesis is to choose appropriate technology for development of rich web applications, along with an appropriate component for visualization of social networks. With this aim, this thesis not only presents different technologies for RIA development, but also defines the use of the GWT tool in great detail. The GWT tool enables simple development of web applications in a single language, Java. It is highly appropriate for development of enterprise web applications, mainly due to the advantage of merging the program code from the server side with the client's side into a common project.

The use of social networks is an ideal way of enabling efficient and intuitive examination of people and connections between them. Visualization of social networks plays an integral role in their research and analysis. It is more than just narrow visualization of interesting images; it is mainly about generation of learnable situations and detection of important information. For this purpose the component Birdeye RaVis, designed with Adobe Flex technology was chosen. RaVis is a powerful and perfected component for outlining graphs or networks, which use XML files as their main data source. With the use of the GWT-Fabridge interface, we made sure that communication between two technologies, as different as GWT and Flex, did not present a problem.

Keywords: RIA, GWT, social networks, RaVis, GWT-FABridge

Uvod

V podjetju Optilab d.o.o razvijamo sisteme za zaznavanje goljufij. Področje, na katerem smo začeli delovati, je zavarovalništvo, natančneje zdravstveno zavarovanje. Kasneje pa smo začeli razvijati še sistem za odkrivanje goljufij v avtomobilskem zavarovalništvu.

Produkt se imenuje Admiral in temelji na večslojni javanski platformi, dodatno pa uporablja tudi JRules, rešitev za upravljanje s poslovnimi pravili, podjetja ILOG. Kot aplikacijski strežnik uporabljamo IBM-ov Websphere Application Server, podatkovna baza pa je prav tako IBM-ova DB2. Izbira Jave za osnovno platformo se je pojavila samodejno, saj večina razvijalcev v ekipi obvlada ta jezik in bi bila torej izbira kakšne druge platforme nesmiselna ter bi povzročila velike časovne zamude na začetku razvoja zaradi prilagajanja neznanemu delovnemu okolju. Tudi izbira aplikacijskega strežnika in podatkovne baze je bila enostavna, saj je IBM-ovo okolje narekoval naš prvi naročnik.

S porastom popularnosti svetovnega spleta, hitrejšimi povezavami in zmogljivejšimi brskalniki se razvija tudi trg spletnih aplikacij, ki postajajo vse bolj raznovrstne in izpopolnjene. Prav to je razlog, da se vse pogosteje spletne aplikacije uporabljajo tudi v zaprtih lokalnih omrežjih oz. v poslovnem svetu, kjer nadomeščajo klasične namizne aplikacije. To je tudi eden od razlogov, da smo se v Optilabu odločili, da novi uporabniški vmesnik predstavimo kot obogateno spletno aplikacijo. Prednosti takih aplikacij je precej, a o tem nekoliko kasneje.

Pri projektu za odkrivanje goljufij v avtomobilskem zavarovanju je eden izmed ciljev izdelati modul, sposoben prikazovanja socialnega omrežja, ki bi pri odkrivanju goljufij v avtomobilskem zavarovanju ponujal učinkovit in intuitiven pregled udeležencev v prometnih nezgodah in povezav med njimi. Na grafičnem vmesniku bi za potrebe učinkovitega preiskovanja prikazovali tudi ostale entitete, na primer avtomobile in dogodke. Modul bi omogočal pogled na izbrano osebo s širše perspektive, kot je to mogoče z opazovanjem izoliranega škodnega primera.

Cilj te diplomske naloge je izbrati primerno tehnologijo oz. koncept za izdelavo obogatene spletne aplikacije ter izdelati ali najti ustrezno komponento za interaktivni prikaz socialnega omrežja, ki bi v čim večji meri ustrezala zahtevam zgoraj omenjenega modula in jo vanjo tudi vključiti.

V nadaljevanju naloge bom v prvem poglavju pojasnil akronim RIA ter za njim v podpoglavjih od 2.1 do 2.3 naštel vodilne ter v podpoglavju 2.4 še ostale tehnologije za razvoj RIA. S pomočjo prednosti in slabosti ter zahtev v tretjem poglavju izberem za naš primer ustrezno tehnologijo. V četrtem poglavju se bom posvetili implementaciji izrisa socialnega omrežja ter v podpoglavju 4.1 utemeljil uporabo GWT za glavno ogrodje naše spletne aplikacije. Za njim je v podpoglavju 4.2 ter 4.3 obrazložen pojem socialnega omrežja in izbira primerne komponente za njen izris.

Zadnje podpoglavje 4.4 je namenjeno vzpostavitvi komunikacije med SWF ter GWT. Kot zaključek pa v poglavju 5 predstavimo še sklepne ugotovitve.

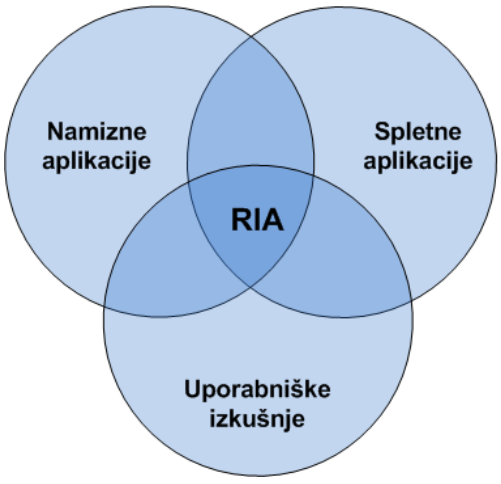
1. Kaj pomeni akronim RIA

RIA ali v slovenščini "obogatena spletna aplikacija" je izraz za spletne rešitve, ki premorejo funkcionalnosti, značilne za namizne aplikacije. Nudi bogate in intuitivne grafične vmesnike, tako da so računalniške aplikacije bolj dostopne in vabljive, zaradi delovanja znotraj spletnih brskalnikov ne potrebujejo posebne namestitve. [1]

Skozi zgodovino programske industrije, smo bili priča kar nekaj večjim spremembam. S centralnega računalnika z "neumnimi" terminali smo prešli na sistem strežnikov in odjemalcev. Uporabniki so tako pridobili na udobnosti uporabe aplikacij in produktivnosti, centralne računalnike pa so pokroviteljsko označili kot zastarele.

Spoznajmo nov izziv, vstopamo namreč v obdobje obogatenih spletnih aplikacij (RIA), ki oživlja moč namiznih aplikacij, vendar znotraj spletnih strani. RIA deluje znotraj navideznega stroja (primer Adobe Flash Player ali Java VM). V nasprotju s preprostim prikazovanjem spletnih strani, naloženih na nekem oddaljenem strežniku, se RIA izvaja na odjemalcu. Večina podatkovnih operacij (sortiranje, grupiranje in filtriranje) je narejenih lokalo kot v starih časih terminalov strežnik odjemalec. Pravi "Déjà vu". V prihodnosti lahko pričakujemo, da bo RIA postala nadomestilo za namizne aplikacije s polnim obsegom funkcionalnosti. Analitiki predvidevajo, da bo v treh do štirih letih večina razvitih projektov v tehnologiji RIA.

RIA združuje prednosti uporabe spleta kot poceni postavitvenega modela z bogatimi uporabniškimi izkušnjami, ki so vsaj tako dobre kot današnje namizne aplikacije. In ker RIA ne zahteva osveževanja celotne strani, je odzivni čas precej krajši in mrežni promet precej manjši.



Slika 1: Koncept RIA tehnologije

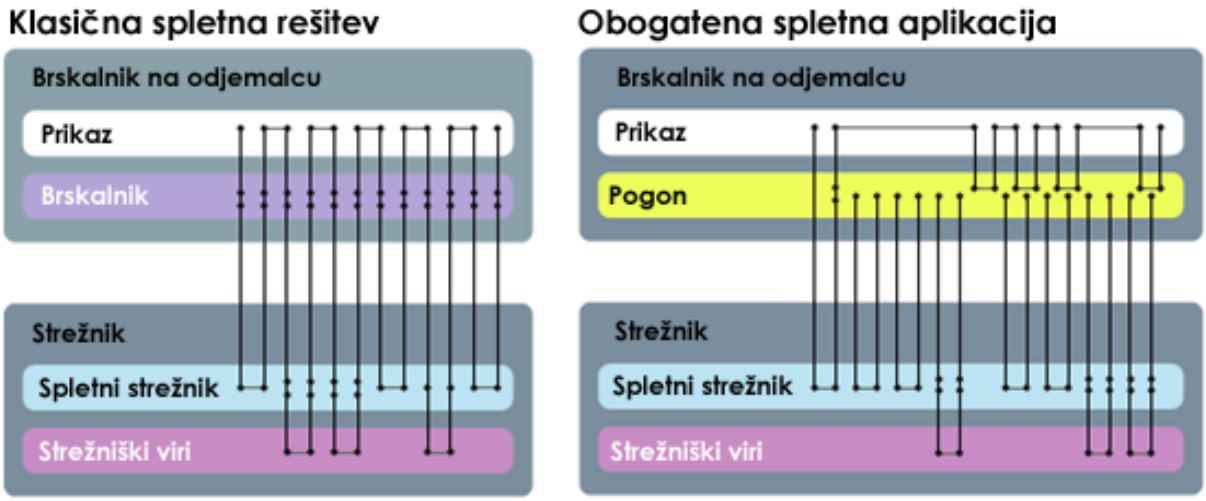
Če ponazorimo razliko med "zastarelimi" oz. statičnimi spletnimi aplikacijami in RIA na primeru spletne trgovine z "nakupovalno košaro". Starejše spletne aplikacije so izdelane s statičnimi stranmi. Ker HTTP protokol ne vzdržuje seje, si ob premiku na drugo stran spletni brskalnik ne zapomni uporabnikovih dejanj na prejšnji strani. Kot

običajna rešitev takšnega problema se na strežniški strani shrani uporabnikovo stanje kot način hranjenja HTTP seje.

Poglejmo primer, kako bi potekala seja v primeru spletne trgovine:

- Uporabnik prične z iskanjem primernega izdelka na spletni strani A.
- Strežnik obdela zahtevo in vrne stran B, ki vsebuje (ali pa tudi ne) iskani izdelek.
- Uporabnik doda izdelek v spletno košaro, kar spet povzroči prenos podatkov do strežnika, kjer se košarica ustvari in shrani na strežniški strani. Nato se strežnik odzove s stranjo C, tako da lahko uporabnik ali nadaljuje z nakupovanjem (ponovi prejšnje tri korake) ali pa nadaljuje k potrditvi nakupa - stran D.
- Pri potrditvi nakupa strežnik prejme izbrane izdelke nakupovalne seje in kot stran E uporabniku pošlje še obrazec o vrsti in načinu dostave. Vnešeni podatki potujejo nazaj na strežnik, kjer se ponovno shranijo in odjemalec prejme stran F z obrazci o načinu in vrsti plačila. Ko so še ti izpolnjeni, se prikaže stran G, kjer uporabnik prejme vnešene podatke ter informativni izračun in končno potrdi svoj nakup.

Najpreprostejši spletni nakup je torej sestavljen iz vsaj sedmih skokov na strežnik. Očitna razlika med namizno aplikacijo in RIA je nekaj sekund za osvežitev strani, čeprav lahko RIA upoštevamo kot dokaj hitro in odzivno spletno aplikacijo. Za tipično spletno aplikacijo je sprejemljiv čas osvežitve do 8 sekund. Ampak, ali bo povprečen uporabnik dovolj motiviran, da nadaljuje nakup? Pomislimo znova, saj mu tak sistem omogoča nekaj sekund, sedemkrat zapovrstjo, da si premisli in opusti oz. prekliče svoj nakup.



Slika 2: Razlika med klasično spletno rešitvijo in RIA

RIA odstrani nepotrebne prehode in v veliki meri izboljša sistemske zmogljivosti tako, da izvede veliko več preračunavanja že na samem odjemalcu, kot to naredi običajno izdelana spletna aplikacija na osnovi statičnih spletnih strani. Poleg tega lahko RIA obravnavamo kot aplikacijo, ki ohranja sejo – kar pomeni, da lahko informacije zbira

že kar na odjemalcu samem. Če poenostavimo, RIA ni nabor strani, ki jih strežnik samo prikazuje. RIA je dejanska aplikacija, ki se izvaja na odjemalcu in v osnovi komunicira s strežnikom, da procesira in izmenjuje podatke.

Tako manjše, strankam namenjene, spletne aplikacije kot tudi obširne poslovne aplikacije so boljše, če so grajene z RIA pristopom. Dobro poznano dejstvo je, da elektronske poslovne spletne strani kot npr. sistem spletne rezervacije vstopnic in spletni trgovci na drobno izgubljajo dobiček, ker uporabniki prehitro zapustijo spletni voziček zaradi neodzivnosti spletne strani med procesom potrditve nakupa. Rezultat takšnih spletnih storitev se velikokrat konča s klici na center za pomoč uporabnikom in s tem velikimi stroški poslovanja. Zmogljivosti, katerega koli sistema, ki za izvajanje potrebuje zaposlene, so kritične za produktivnost podjetja in RIA zagotavlja performančni pospešek nad običajnimi HTML aplikacijami, medtem ko zmanjšuje operativne ter infrastrukturne stroške.

Macromedia, podjetje za oblikovanje in razvoj spletnih aplikacij, je razvilo produkt z imenom Macromedia Flash ter razvojno orodje Macromedia Dreamweaver. Leta 2002 je Macromedia prvič predstavila izraz "obogatena spletna aplikacija", ko je kot odgovor žalostnega stanja "zastarelega" spletnega omrežja, ki ga danes poznamo kot web 1.0 predstavljala svoje nove storitve. Te so kasneje prešle v nadzor podjetja Adobe Systems, ki je konec leta 2005 dokončno prevzel podjetje Macromedia.

Vendar pa so se prve RIA aplikacije rodile že leta 1995, ko je bila narejena Java. Uvodna popularnost Jave se je odrazila kot val manjših programov z imenom java applets, ki so jih uporabniki prenašali s spleta in nameščali v brskalnike. Apleti ustvarjeni z AWT (pozneje s swing-om) in knjižnice so poganjali brskalnikov navidezni stroj (virtual machine).

Obogatene spletne aplikacije združujejo koristi uporabe spleta kot cenovno ugodno možnost postavitve modela z bogatimi uporabniškimi izkušnjami, ki so vsaj tako dobre kot večina današnjih namiznih aplikacij. In ker RIA ne zahteva osveževanja celotne strani za prikaz podatkov, je odzivni čas precej krajši ter omrežna obremenitev precej manjša. Predstavljamo si jo lahko kot globalno dosegljivo odjemalec/strežnik aplikacijo.

2. RIA platforme: vodilne tehnologije

Verjetno ni težko ugotoviti, da obstaja več kot samo en način, da razvijemo RIA. Če na hitro naštejemo nekaj vodilnih produktov in tehnologij ter njihovih lastnosti:

- **Flex**, s katerim lahko izdelamo ActionScript aplikacijo za Flash Player. Ta visokozmogljivi multimedijski virtualni pogon, ki poganja kodne datoteke v SWF formatu (izgivorimo kot swif), je tako razširjen, da ga jemljemo že kot standardno opremo brskalnika. Predvajalnikov JIT (Just In Time) prevajalnik prevede SWF kodo v izvorno strojno kodo za hitrejše izvajanje. Gre za "pripomoček", ki je specifičen za Flex. Čeprav prve verzije Flex-a obstajajo že od leta 2004, le-te na začetku niso podpirale JIT prevajanja.
- **Java**, kjer programer lahko razvija Java applete. Kot omenjeno – gre za rešitev, ki je na voljo že od leta 1995.
- **WPF** (Windows Presentation Foundation), Microsoftova rešitev, ki je bila izdana kot del .NET 3.0 novembra 2006 in jo lahko uporabimo za razvoj obeh, tako namiznih kot tudi spletnih aplikacij.
- **Silverlight**, ki je nadgradnja WPF, ponuja spletno aplikacijsko ogrodje z razvojnim okoljem podobnim Flex-ovemu. Ponuja dodatne interaktivne zmogljivosti ter podporo .NET programskim jezikom in razvojnim orodjem.
- **AJAX**, poznan tudi kot DHTML, je nastal okrog leta 1998. Ta rešitev je bila nedavno pospešena z *XMLHttpRequest* API podporo za vse glavne brskalnike. AJAX je služil kot klic prebujenja za uporabnike in razvijalce. Najpogosteje je to prvi korak pri prehodu iz zastarele spletne tehnologije v svet RIA, navkljub resni pomanjkljivosti, kot je skrb za podporo brskalnikove nekompatibilnosti in slabemu programskemu modelu.
- **GWT** kot odgovor na AJAX-ove pomanjkljivosti in podpora Java programerjem, da svoje znanje Java programskega jezika uporabijo pri razvoju "novodobnih" RIA zahtev.

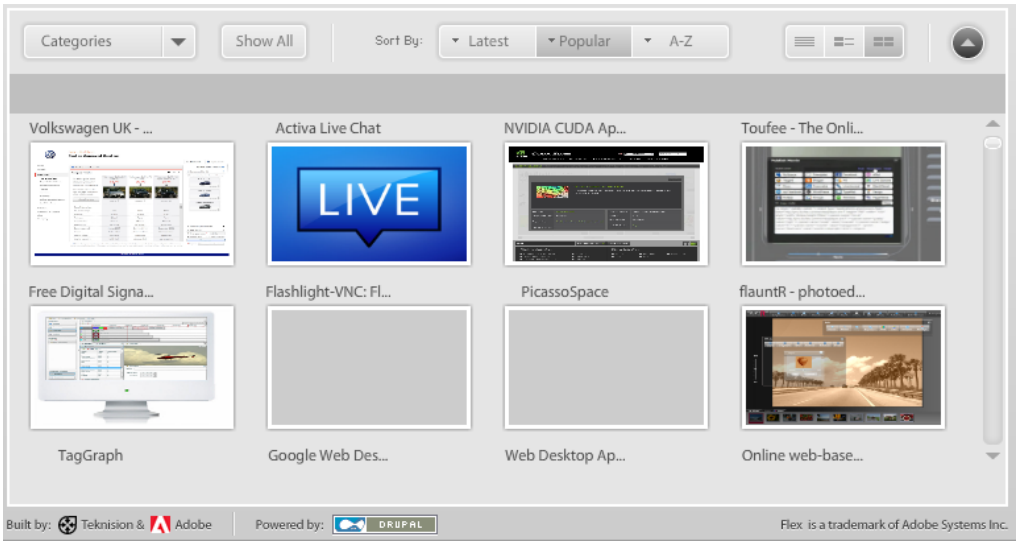
V sledečih poglavjih bomo podrobneje pregledali, kaj nam naštete tehnologije ponujajo ter kakšne so prednosti in slabosti uporabe posameznih tehnologij.

2.1. Adobe flex

Flex 3 aplikacije so platfomno neodvisne in se izvajajo znotraj Flash predvajalnika, ki ga lahko označimo kot lahek virtualni pogon. Flex razvojni paket vsebuje:

- XML osnovan jezik imenovan MXML, ki omogoča programiranje GUI komponent.
- Standardni objektno usmerjen programski jezik, ActionScript 3.0 osnovan po zadnjih ECMA Script specifikacijah.
- Strežniško integracijo preko FDS (Flex Data Services), ki ponuja aplikacijam na odjemalčevi strani dostop do J2EE sveta.
- Že izdelane grafične komponente, dostop do multimedijskih kontrol itd., za katere je potrebna plačljiva licenca.
- Na Eclipse-u osnovano razvojno okolje s polnim naborom funkcij, podporo ter avtomatsko postavitvijo, razhroščevalnikom ter sledenjem, za katerega je prav tako potrebna plačljiva licenca.

Flex 3 platforma je preprosto razširljiva in se dobro integrira z Javo, ColdFusion, PHP, Ruby, ASP in podobnimi tehnologijami. Adobe Flex in Adobe Air (prej poznan kot projekt Adobe Apollo) omogočata gradnjo namiznih in spletnih aplikacij, ki se kot SWF datoteke izvajajo v Flash predvajalniku.



Slika 3: Primeri spletnih aplikacij razvitih z Adobe Flex tehnologijo

SWF je odprti format in zato obstajajo tudi zunanji odprtokodni produkti, ki ponujajo orodja za razvoj RIA ponujen preko Flash predvajalnika kot npr. OpenLaszlo od Laszlo Systems.

Proces za izdelavo osnovne Flex aplikacije je z razvijalskega stališča dokaj preprost in je sestavljen iz naslednjih korakov:

1. Zasnova aplikacije s sestavo MXML komponent. Uporaba Flex razvojnega okolja omogoča uporabniku prijazen način grafičnega oblikovanja s sestavljanjem komponent na način povleci in spusti. Lahko pa tudi kodo ročno zapišemo v MXML.
2. Poslovno logiko in ostalo funkcionalnost zapišemo z ActionScript kodo.
3. Kodo prevedemo. Flex prevajalnik avtomatično prevede MXML v ActionScript in ustvari strojno kodo (bytecode) v obliki SWF datoteke, ki jo poganjamo v Flash predvajalniku. Če uporabljamo Flex razvojno okolje se ta proces izvede samodejno.
4. Na izbran strežnik postavimo SWF datoteko skupaj s spremljajočimi HTML datotekami. Postavitev ter izdelava je popolnoma transparentna, v kolikor uporabljamo Flex razvojno okolje.

Naprednejše Flex aplikacije lahko vključujejo interakcijo s strežniškimi sistemi preko FDS (Flex Data Services), ki zagotavlja oddaljeni dostop do strežniških Java objektov ter JEE komponent, razširjenih sporočilnih podpor (vključno z JMS integracijo), sinhronizacijo s podatkovno bazo in integracijo z ostalimi tehnologijami.

2.2. Java Applet tehnologija

Čeprav je programski jezik Java postal popularen predvsem zaradi appletov, pa ti niso postali glavni del Javanskega sveta. Glavni razlog: velika poraba delovnega spomina, ki ga je zahteval JVM (trenutno 16 MB).

Obstaja sicer nekaj poskusov, da bi zmanjšali velikost JVM uporabljene v spletnih brskalnikih in Java Browser Edition Projekt sedaj potrebuje "le" kakšnih 3 MB za izvajanje primitivnega "Pozdravljen svet" appleta. Vendar se to ne more primerjati s Flash predvajalnikom, ki je uspel prilagoditi dva navidezna stroja v 1,2 MB velikem prenosu, ki lahko poganja katerokoli RIA, ne glede na kompleksnost.

Dodatna pomanjkljivost Java appletov je, da ne ponujajo samodejnega prenosa prave verzije JVM skupaj z appleti. Flash predvajalnik express install dela točno to.

Ali pa vzemimo primer avdio in video integracije. Danes je že popolnoma običajno, da imamo avdio in video komponente ugnezdene v spletne strani. Vendar pa multimedijski Java API še vedno ostaja neroden za uporabo in prikaz avdio ter video vsebin.

Na voljo obstaja tudi Java GUI ogrodje, imenovano Java Swing, ki pa se v javnosti ni preveč dobro oprijel, pa čeprav je bil mišljen kot platformsko neodvisen, preprost, lep, kompakten MVC GUI gradnik. Java Swing je zelo zrela in robustna tehnologija za razvoj GUI aplikacij, dostavljenih preko spleta ali pa nameščenih na namizju. Z njo lahko naredimo praktično vse, če si lahko privoščimo. S tem pa tukaj ne mislim na plačevanje dragih licenc, ampak na daljši razvojni cikel in potrebe po vključevanju

izkušenih programerjev. Zaradi tega je izvedba Swing projektov običajno precej draga za izgradnjo ter vzdrževanje.

2.3. Microsoft WPF

Microsoftov Windows Presentation Foundation oz. WPF, je postal na voljo z izdajo Viste. Uporablja na XML-ju osnovan deklarativni programski jezik imenovan XAML, s katerim razvijemo grafični uporabnški vmesnik, za poslovno logiko pa C# kot univerzalni jezik. WPF je primeren za ustvarjanje tako RIA kot namiznih aplikacij. XBAP pomeni XAML Browser Application in način za ustvarjanje RIA, ki se izvajajo v Internet Explorer-ju.

Za razvoj WPF aplikacij lahko razvijalci uporabijo Microsoft-ov Visual Studio z nameščenim .NET 3.+ dodatkom. WPF razvijalci uporabljajo isti jezik za pisanje XBAP in namiznih aplikacij: ker kodni odseki niso dovoljeni v XBAP se programska koda umesti v *ifdef* blok.

Microsoftova XAML koda je precej podobna Adobe-ovemu MXML. Čeprav je danes Flex precej bogatejši kot WPF je Microsoft vzpostavil svojo razvijalsko knjižnjico, Adobe pa je bolj naklonjen oblikovalcem.

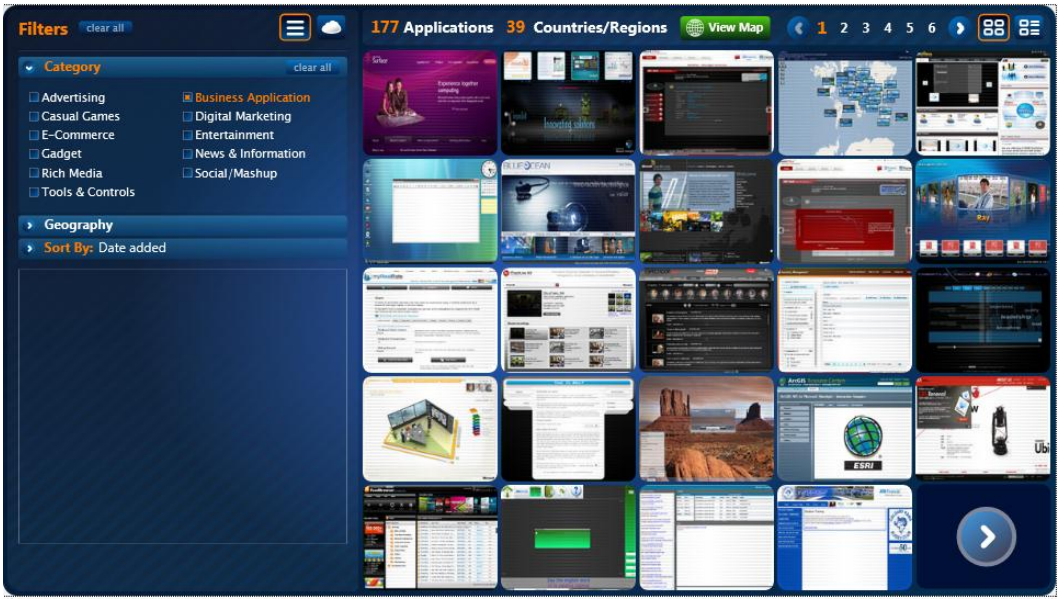
2.1. Microsoft Silverlight

Microsoft je nedavno izdal tudi Silverlight (projektno ime WPF/E). Microsoft Silverlight je spletno aplikacijsko ogrodje (web application framework), okolje podobno Adobe Flash-u. Verzija 2, izdana oktobra 2008, je prinesla dodatne interaktivne izboljšave ter podporo .NET jezikom ter razvojnim orodjem. Zadnja verzija 3.0 (beta) pa je na voljo od 18. Marca 2009. Zadnja stabilna verzija je bila izdana 9. Julija 2009.

Silverlight je združljiv z raznimi brskalniki, ki se uporabljajo v operacijskih sistemih, kot so Microsoft Windows in Mac OS X. Mobilne naprave, začenši z Windows Mobile 6 ter Symbian-om (serije 60), bodo najverjetneje podprte z letom 2010. Brezplačni implementacijski program, ki ga je izdelal Novell v sodelovanju z Microsoftom in nosi ime Moonlight, je na voljo, da ponudi možnost kompatibilne fukncionalnosti v operacijskem sistemu Linux.

Silverlight uporablja grafični sistem, ki je podoben WPF in združuje grafiko, animacije, zvok ter interaktivnost v enotno izvajalno okolje. V Silverlight aplikacijah so uporabniški vmesniki deklarirani z XAML, za poslovno logiko pa se uporablja .NET ogrodje. XAML se lahko uporablja kot označevanje vektorske grafike in

animacije. Tekstovno vsebino, ustvarjeno s Sliverlightom, lahko iščemo in indeksiramo z iskalniki, ker se ne prevaja ampak je predstavljena kot besedilo (XAML). Prav tako se Silverlight lahko uporablja za izdelavo stranskih gradnikov v operacijskem sistemu Microsoft Vista.



Slika 4: Primeri spletnih aplikacij razvitih z Microsoft Silverlight tehnologijo

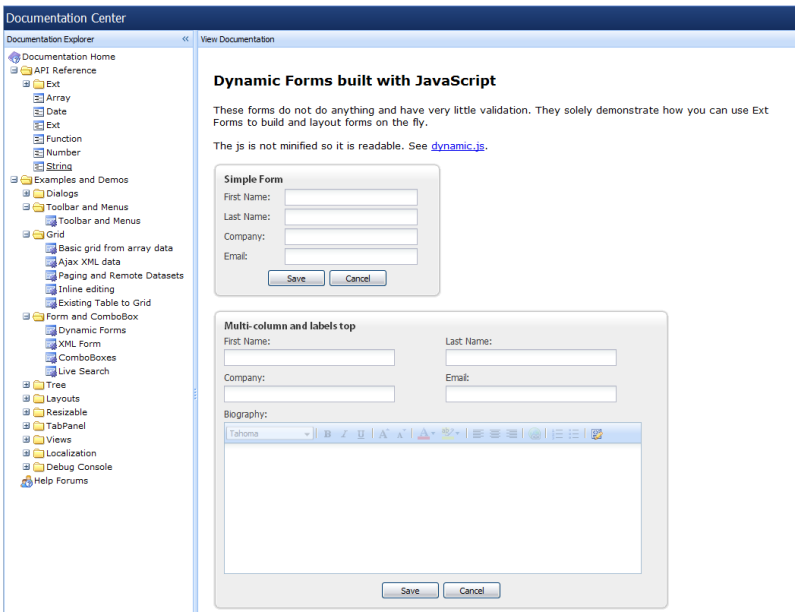
Silverlight podpira predvajanje VMW, VMA ter MP3 formate medijskih vsebin preko vseh spletnih brskalnikov brez zahtev po Windows Media predvajalniku, Windows Media predvajalnika ActiveX kontrole ali Windows Media spletnega vtičnika. Silverlight omogoča tudi dinamično nalaganje XML vsebine, ki jo lahko upravljamo preko DOM vmesnika, tehnika ki je ustaljena pri običajnih AJAX tehnikah. Silverlight ponuja tudi *Downloader* objekt, ki se lahko uporablja za prenos vsebin, kot npr. skript, zvočnih in video posnetkov, ali ostalih podatkov, kar bi lahko zahtevali z aplikacijo. Od verzije 2 je lahko programska logika zapisana v kateremkoli .NET jeziku, vključno z nekaterimi izpeljankami običajnih dinamičnih programskih jezikov kot IronRuby in IronPython.

2.2. AJAX

Kot DHTML in LAMP, AJAX ni samostojna tehnologija ampak je skupek več tehnologij. Koncept AJAX uporablja naslednjo kombinacijo :

- HTML in CSS za označevanje ter oblikovanje.
- DOM dostopen preko JavaScript-a za dinamičen prikaz ter interakcijo.
- Kot metoda za izmenjavo podatkov med brskalnikom ter strežnikom se največkrat uporablja XMLHTttpRequest (XHR), občasno pa tudi IFrame z dinamično dodano <script> značko (tag).
- Za oblikovanje podatkov, ki se pošiljajo do brskalnika, se pogosto uporabljajo XML, predoblikovani HTML, navadno besedilo in JavaScript Object Notation (JSON). Ti podatki se lahko generirajo dinamično z izvajanjem skript na strežniški strani.

Izraz AJAX je skoval Jesse James Garret februarja 2005 in, kot sem že omenil, delno izvira iz asinhronega klica *XmlHttpRequest*, ki ga je implementirala Mozilla. Vendar so se zametki uporabe AJAX tehnologije pojavljali že leta 1999, ko je veliko razvijalcev uporabljalo Microsoftovo verzijo *XMLHttpRequest* klica in alternativne tehnike kot npr. IFrame. Te tehnike olajšajo sinhrono in asinhrono komunikacijo med skripto na odjemalčevi strani ter strežniško kodo.



Slika 5: Primeri uporabe AJAX gradnikov

Medtem ko se ostale rešitve za izdelavo RIA, ki sem jih že omenil, osnovane na trdnih temeljih virtualnih strojev, AJAX ne premore lastnega navideznega stroja, zato vsak brskalnik implementira AJAX-ove gradnike drugače. Obstaja možnost, da bo pri postavitvi AJAX aplikacije, za pravilno delovanje v različnih brskalnikih, kjer naj bi se aplikacija izvajala, potrebna sprememba kode. To pa zna biti velika slabost pri izbiri te tehnologije.

Glede na povedano, internetni giganti kot Google, Yahoo, in Amazon gradijo AJAX knjižnice in ogrodja kot Google Web Toolkit (GWT). Zaradi "nezrele" tehnologije so ta ogrodja potrebna stalne pozornosti ter dopolnitev, če pride do morebitnih sprememb knjižnic.

2.2.1. Pomanjkljivosti AJAX koncepta

Možnost izdelave dinamičnih spletnih aplikacij brez nakupa dodatnih programov je AJAX-ova največja privlačnost. Zagotovo smo že vsi slišali za izraz:"AJAX je brezplačen". Če te besede malce obrnemo, lahko poenostavimo, da nobeno komercialno AJAX orodje ni dovolj vredno, da ga kupimo. Obstaja na stotine knjižnic, orodij ter krmilnikov, ki dajejo vtis, da je razvoj AJAX aplikacij hiter, poceni ter strateško varen, ker smo pri razvoju svobodni in nismo vezani na nobenega proizvajalca programske opreme. Vendar pa lahko kaj kmalu ugotovimo, da temu ni tako, saj ne bomo ročno pisali celotne JavaScript kode, in bomo zaradi tega izbrali že izdelano AJAX knjižnico nekega proizvajalca. Torej pomislimo: če razvijamo iz nič, potrebujemo komunikacijski nivo, sporočilne in usmerjevalne mehanizme, Http vohljače, knjižnico z uporabniškimi vmesniki, deljenimi objekti ter dogodkovnimi modeli, grafično razvojno okolje, ki razume in prepozna te komponente, ter razhroščevalnik. Po vrhu vsega je tu še podpora večjezičnosti, dostopnost za omejene, ter podpora avtomatskim orodjem za preverjanje in testiranje. Ko vse to seštejemo skupaj, ugotovimo, da je svoboda glede neodvisnosti od proizvajalcev programske opreme zelo omejena.

Če se torej postavimo na realna tla, lahko ugotovimo, da dolgotrajen razvojni cikel, pomanjkanje brezplačnih in kvalitetnih GUI komponent ter spodnji seznam pomanjkljivosti prikažejo AJAX kot manj privlačen ter celo drag način razvijanja RIA.

Če naštejemo nekaj trenutnih pomanjkljivosti:

- Preprostost uporabe AJAX klicev pogosto dramatično poveča število uporabniško generiranih poizvedb na spletne strežnike in njihove baze. To lahko vodi k slabšim odzivnim časom in dodatni strojni opremi za podporo AJAX-ovih vmesnikov.
- Dinamično ustvarjene strani z uporabo zaporednih AJAX poizvedb se samodejno ne vpisujejo v zgodovino brskalnika. Kar pomeni, da se ob kliku na brskalnikov gumb "prejšnja stran" ne bo prikazala prejšnja, ampak prva prikazana stran. Nadomestna rešitev obstaja z uporabo *IFrame*-ov, ki sprožijo spremembe v brskalnikovi zgodovini ter postavijo sidro na URL (sledeč # znak), kadar se AJAX izvaja ter spremlja spremembe.
- Dinamične spletne strani so prav tako težavne za uporabnike, ki želijo stran v aplikaciji dodati med svoje zaznamke. Rešitve za ta problem obstajajo, mnoge uporabljajo *URL Fragment Identifier* (del URL-ja za # znakom) za sledenje ter možnost uporabnikov, da se vrnejo na želeno stran/stanje v aplikaciji.
- Ker večina spletnih pajkov ne izvaja JavaScript kode, je težavno indeksiranje ter umeščanje strani v iskalne programe. To pa pomeni, da bi morale javne indeksirne spletne aplikacije poskrbeti za alternativo, ki bi omogočala avtomatsko indeksiranje ter vsebinsko iskanje AJAX spletnih strani.

- Uporabnik, katerega brskalnik ne omogoča AJAX-a ali JavaScripta, ali pa ima JavaScript onemogočen, ne bo mogel uporabljati njegove funkcionalnosti. Prav tako Mobilni telefoni, dlančniki ter ostale mobilne naprave, ki nimajo podpore JavaScripta, ne bodo pravilno prebrale dinamično generiranih vsebin.
- AJAX je s stališča varnosti lažje ranljiv za napad hekerjev in zlonamerne kode. Dejstvo je, da je bilo nedavno dokazano, da je črv ukradel kup naslovov iz Yahoo adresarja.

Dion Hinchcliffe, glavni urednik revije AJAXWorld, je naštel nekaj pomembnih stvari, ki bi jih moral vedeti vsak AJAX razvijalec:

- Brskalniki nikoli niso bili namenjeni AJAX-u.
- Ne potrebujemo toliko Web service-ov kot mislimo.
- AJAX je bolj zapleten kot tradicionalno spletno oblikovanje ter razvijanje.
- AJAX-ova orodja in gradniki so še vedno v nastajanju in do danes še ni vodilnih.
- Dobre AJAX programerje je težko najti.
- Potrebno je veliko dela, da obidemo vse AJAX-ove pomanjkljivosti.
- AJAX je le en element uspešne RIA strategije.

Hinchliffe tudi pravi: "Dodatki RIA platform, kot so Flex, OpenLaszlo in Silverlight RIA strategiji je praktično nujna za temeljit izkoristek obsežnih zmogljivosti, za katere želimo, da jih imajo aplikacije. To se izkaže za resnično predvsem okoli podpore obogatenih spletnih vsebin, kot so avdio in video – ki je AJAX praktično nima – pa tudi okrog takšnih osnovnih zadev, kot je podpora tiskanju. To so stvari, pri katerih bolj razvite Flash osnovane RIA platforme resnično zasijejo in se jih lažje sprogramira. Seveda pa se bo AJAX pospešeno razvijal, še posebno, ko bo njegova hrbtenica poskrbela za dovršeno podporo omenjenim stvarem.

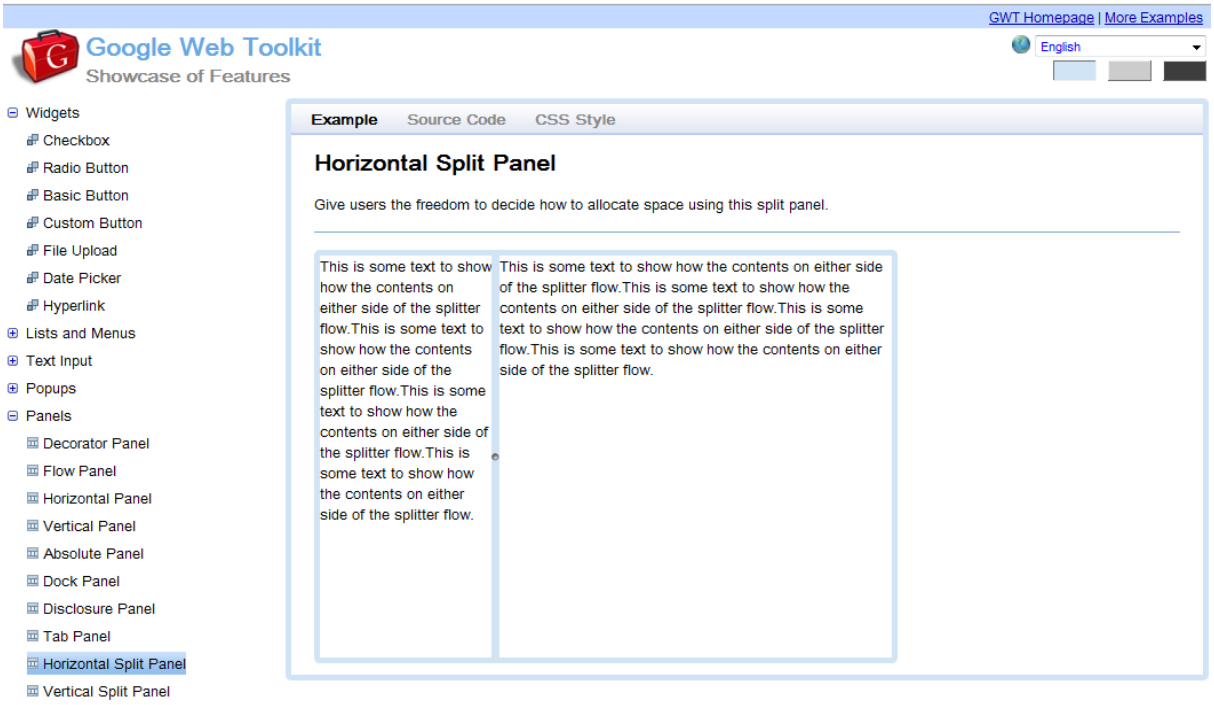
Zanimivo pa je, da sama prihodnost AJAX poslovnih aplikacij ni tako žalostna, kot se zdi na prvi pogled, kar bom pojasnil naknadno.

Če povzamemo, za razvoj nove poslovne RIA iz nič, AJAX morda ni prava rešitev. Verjetno je bolje izbrati stabilno razvojno okolje, ki vsebuje navidezni stroj v skupnem času delovanja kot npr. Flex/Flash, Java/GWT ali Silverlight. Katero koli okolje od teh omogoča lažji razvoj kot AJAX. Če pa že imamo delujočo AJAX aplikacijo, jo lahko preprosto integriramo z novo Flex RIA v obstoječo AJAX aplikacijo s pomočjo vmesnikov, kot so FABridge od Adobe, ki skrbi za komunikacijo med AJAX-om ter Flex-om.

2.3. GWT

Kratica GWT pomeni Google Web Toolkit. Gre za odprtokodni nabor orodij, ki omogočajo spletnim razvijalcem razvijanje ter vzdrževanje JavaScript osprednjih (front-end) aplikacij v Javi. Razen nekaj osnovnih (native) knjižnic je vse ostalo javanska koda. [2]

GWT omogoča pisanje programov v javi, ki se avtomatsko prevedejo v JavaScript tako, da jih lahko postavimo kot AJAX aplikacije. To ni prvi poskus orodja, ki ponuja prevajanje Jave v JavaScript. Vzemimo za primer odprtokodni projekt Java2Script. Gre za Eclipse vtičnik, ki implementira Eclipse SWT knjižnico v JavaScriptu. Vendar pa je GWT-jev uspeh dodaten dokaz, kako pomembno je imeti podporo velikega komercialnega podjetja, kot je Google, za kakršno koli AJAX iniciativo.



Slika 6: Primeri uporabe GWT gradnikov

Glavne GWT komponente:

- GWT Java-to-JavaScript prevajalnik, ki prevaja Java programski jezik v JavaScript skriptni jezik.
- GWT gostujoči spletni brskalnik omogoča razvijalcu zagon ter izvajanje GWT aplikacij v gostujočem načinu. Program izvaja Javo znotraj JVM brez prevajanja v JavaScript, kar omogoča lažje razhroščevanje in sledenje.
- JRE emulacijske knjižnice: gre za JavaScript implementacije pogosto uporabljenih razredov v standardni javanski knjižnici, kot so večina razredov iz *java.lang* paketa ter skupek razredov iz *java.util* paketa.

- GWT Web UI knjižnica razredov: množica prirejenih vmesnikov in razredov za ustvarjanje gradnikov (widget-ov).

Zanimiva lastnost GWT tehnologije je, da lahko prevaja Javo v različne verzije JavaScripta in tako omogoča podporo različnim brskalnikom. To pomeni, da je GWT sposoben prevesti Javo v JavaScript tako, da bo le-ta deloval enako v vseh brskalnikih.

2.4. Ostale RIA rešitve

2.4.1. OpenLaszlo

OpenLaszlo podjetja LaszloSystems je odprtokodni produkt s CPL licenco. Laszlo je zasnoval David Temkin jeseni leta 2001. Projekt pa je, zanimivo, dobil ime po mačku grafičnega oblikovalca in soustanovitelja LaszloSystems Petra Andree. [**3**]

Laszlo aplikacije lahko na strežnik postavimo kot običajne Java servlete, ki se prevedejo in brskalniku dinamično vrnejo vsebino. Za to je potrebna uporaba OpenLaszlo strežnika.

Alternativno lahko Laszlo aplikacije prevedemo iz LZX jezika v DHTML ali pa binarne SWF datoteke, ki se predvajajo s pomočjo Flash predvajalnika. Ta metoda je znana kot SOLO strežniška postavitev. Takšnim postavitvam pa manjka nekaj funkcionalnosti, kot so zmožnost uporabe SOAP spletnih storitev ter Javinih oddaljenih procedurnih klicev (Remote Procedure Calls).

Ena izmed Laszlovih priljubljenih lastnosti, poleg tega da gre za odprtokodni produkt, je, da lahko z njim izdelamo aplikacije, ki tečejo v Flash predvajalnikih od verzije 6 naprej. Vendar pa razširjenost novejših verzij Flash predvajalnikov nekako izničuje to prednost.

2.4.2. Nexaweb

Nexaweb ponuja lahki odjemalec, osnovan na Javi, ki ne potrebuje dodatnih namestitev na namizje. Stanje aplikacije se nadzoruje z manjšim Java appletom, ki se izvaja na odjemalcu. Ta applet komunicira z Nexaweb Java EE aplikacijo, kadar je to potrebno. Da bi se izognili "težavam" pri verzijah Jave JRE, ki je nameščena po spletnih brskalinikih, uporablja Nexaweb JRE 1.1 verzijo za svoj applet. Ta je namreč podprta v vseh glavnih brskalnikih. Ta applet deloma posodobi spletno stran, ko se stanje aplikacije spremeni. Uporabniški vmesnik je definiran z uporabo deklarativnega XML jezika, aplikacijska logika pa se programira v Javi. Za razvoj se uporablja grafični urejevalnik podoben Eclipse.

2.4.3. Canoo

Canoo ponuja tako imenovani ultralahki odjemalec. Gre za zbirko orodij ter ogrodje za izdelavo RIA v Javi. Preko knjižnic je na voljo tudi podpora za Swing znotraj tako imenovane "lahki odjemalec" arhitekture.

Razvijalec uporablja strežniške proxy razrede, ki so podobni aplikacijskemu programskemu vmesniku v Swing-u. Knjižnica poskrbi za ločitev med odjemalcem in strežnikom, vključno s sinhronizacijo in komunikacijo teh dveh polovic z uporabo lastniškega protokola (Proprietary Protocol). Manjši prezentacijski pogon teče na odjemalcu, medtem ko se aplikacija izvaja na strežniku

Glavna ideja ultralahkih odjemalcev je izvajanje aplikacij na centralno nadzorovanem spletnem strežniku, podobno kot HTML osnovane aplikacije. Le prezentacijski pogon neodvisnih aplikacij se izvaja na namizju.

2.4.4. Backbase

Backbase vključuje zelo lahek pogon, ki je v celotin napisan v JavaScript-u. Neopazno se naloži na začetku odjemalčeve seje in na odjemalcu poskrbi za prezentacijsko ogrodje, ki vključuje orodja za vzdrževanje seje na odjemalcu, sinhronizacijo med odjemalcem in strežnikom ter branje podatkov. Prezentacijsko ogrodje podpira urejanje preko povleci in spusti metode, grafično oblikovanje ter povezovanje s podatkovnimi viri. Backbase se je oddaljil od aplikacij, grajenih s statičnimi stranmi in nam ponuja tako imenovan vse-na-eni-strani vmesnik. Ponuja nam knjižnico z več kot 50 že izdelanimi GUI komponentami napisanih v DHTML, JavaScript, CSS ter XML jezikih.

3. Izbira primerne tehnologije

V prejšnjih poglavjih naštete tehnologije še zdaleč ne tvorijo popolnega seznama RIA tehnologij in pristopov. Nove tehnologije pridejo in grejo. Samo nekatere od njih so dovolj dobre in zanimive, da se uveljavijo ter ostanejo v izboru razvijalcev in programerjev, ki jih uporabljajo za razvoj poslovnih aplikacij. Vse te tehnične podrobnosti, dodatki in zanimivosti so pomembni, vendar pa to ni vse, kar zagotavlja uspeh.

Ena glavnih skrbi kateregakoli razvojnega vodje je imeti na voljo veliko razvijalcev, ki dobro obvladajo enega ali več določenih programskih jezikov. Vemo tudi, da je na trgu delovne sile veliko, če ne največ programerjev, ki se dobro znajdejo v Java svetu. Razvijalcem, ki imajo izkušnje pri razvijanju Swing ali pa Servlet/Java EE aplikacij tudi Flex ne bo tuj. Še bolj pa se bodo znašli v GWT. Eclipse, kot razvojno okolje, Java razvijalcem ne sme biti tuje, kar pomeni, da pri GWT ne bo težvav, pri Flex-u pa bodo tudi kaj kmalu našli velike podobnosti. To pa pomeni, da bo učna krivulja teh dveh novih tehnologij dokaj strma.

Druga pomembna skrb pa je varnost aplikacij, ki jih bomo razvijali. Kruta resničnost je, da so JavaScript aplikacije dokaj ranljive pri več vrstah varnostnih pasti, v kolikor seveda razvijalci pri razvoju niso pozorni. Ker GWT proizvaja JavaScript kodo, isto velja tudi za GWT razvijalce. In ker je namen GWT orodja omogočiti razvijalcem, da se osredotočijo predvsem na želje njihovih uporabnikov, namesto da bi se ukvarjali s problemi JavaScript-a in brskalnikov, se precej hitro zgodi, da na to pozabimo. Seveda so rešitve dokumentirane znotraj GWT dokumentacije. Vsekakor pa vseeno lahko pričakujemo izboljšave, saj je cilj razvijalcev izdelovati tako varne kot tudi funkcionalne aplikacije.

Tudi pri aplikacijah, grajenih s Flexom, obstaja kar nekaj izzivov pri vzpostavljanju varnosti, saj le-ta podpira več različnih protokolov, tako da mora biti Java EE varnost, ki se nanaša predvsem na HTTP sejo, razširjena, vendar gre tudi tu za dokaj preprost in dobro dokumentiran proces.

Slika 7: Izbira primerne RIA tehnologije

Ob vseh naštetih prednostih pa tudi pomanjkljivostih je vendarle nekako potrebno sprejeti odločitev ter sprejeti tehnologijo oz. koncept, ki naj bi bil primeren za izdelavo želene aplikacije. Ob vsem naštetem in dejstvu, da je v Optilabu zaposlenih največ Java programerjev, sem se odločil za uporabo orodja GWT, ki ga bomo kot ogrodje uporabili za našo osprednjo (front-end) aplikacijo.

GWT je orodje, ki je zelo primerno za razvoj poslovnih spletnih aplikacij, predvsem zaradi prednosti združevanja programske kode na strežniškem delu ter odjemalčevi strani v skupen projekt. Ker je razvojni jezik za to orodje Java, se pri njem odražajo naslednje prednosti:

- Obstaja večje število Java programerjev kot JavaScript programerjev.
- Večje je število orodij, knjižnic ter podpore za Java razvoj.
- Gre za podoben način razvijanja, kot ga poznamo pri razvoju namiznih aplikacij.
- Več je že vnaprej pripravljenih naprednih gradnikov za spletne strani.
- Olajšano je odpravljanje napak pri razvoju z uporabo razhroščevanja ter sledenja.

Z izbiro ustrezne tehnologije je tako izveden prvi del naloge, ki smo si jo zadali. Zato se v prihodnjih poglavjih lahko osredotočimo na izzive pri implementaciji izrisa socialnega omrežja

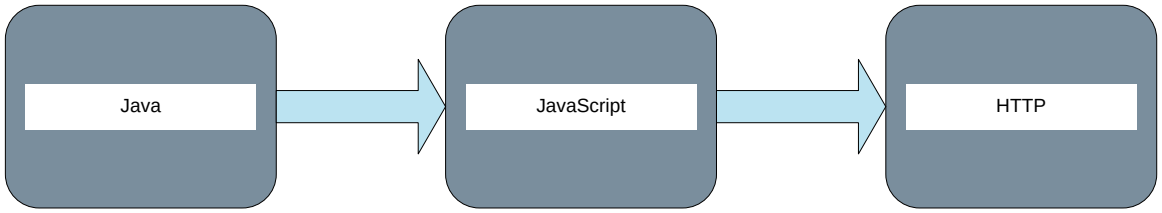
4. Implementacija izrisa socialnega omrežja

Za dokončno izvedbo aplikacije z integriranim modulom za vizualizacijo socialnega omrežja v grobem potrebujemo tri korake. Prvi je izdelava ogrodja, kamor bomo izbrani modul umestili. Za ogrodje bomo uporabili izbrano tehnologijo GWT.

4.1. Uporaba GWT kot glavno ogrodje

S pisanjem kode v enotnem jeziku tako za strežnik kot odjemalec lahko to kodo uporabljamo na obeh straneh. Tako lahko že izdelane gradnike, kot so npr. validacija vnosnega polja, uporabimo tako na strežniški strani kot na strani odjemalca.

Razvijalcem v GWT-ju ni potrebno skrbeti za različno interpretacijo JavaScript kode na različnih spletnih brskalnikih, saj za to poskrbi kar GWT sam. Sicer so razlike v interpretaciji majhne, a pomembne za pravilen prikaz ter izvajanje naše spletne aplikacije. Tako so programerju prikrajšane skrbi za kompatibilnost ter težave pravilnega delovanja na dnevno novih spletnih brskalnikih.



Slika 8: GWT pristop: izvorna koda v Javi se prevede v JavaScript, ki se v brskalniku izvede kot HTML/JavaScript/CSS

4.1.1. Kaj potrebujemo za razvoj GWT aplikacij?

Ker je Java platformno neodvisna, lahko v GWT-ju razvijamo spletne aplikacije na različnih operacijskih sistemih, kot so Windows, Unix, Linux in ostali. Potrebujemo le GWT in Javo. Najprimernejši Java paket za razvoj v GWT je JDK, ki je tako kot GWT brezplačen in na voljo na Sun-ovi spletni strani.

Za pisanje kode oz. razvoj lahko uporabljamo kateri koli urejevalnik besedila, vendar pa je za razvijalca veliko lažje ter časovno ugodnejše, če za razvoj večjih spletnih aplikacij uporablja Eclipse razvojno orodje. Pri samem Eclipse-u je vredno omeniti tudi to, da je na voljo precej vtičnikov, ki še bolj olajšajo ter pohitrijo sam razvoj. Eden takšnih je zagotovo Google Plugin for Eclipse, ki je za prenos na voljo na

uradni spletni strani projekta GWT. Prav tako je tudi GWT, od verzije 1.7 naprej, možno namestiti kot Eclipse vtičnik (Google Plugin for Eclipse).

4.1.2. Kreiranje GWT projekta

Za tiste, ki se odločijo za razvoj s preprostim razvojnim okoljem, je v GWT za pomoč pripravljena skripta *webAppCreator*, s katero lahko kreiramo vse datoteke, ki jih bomo potrebovali za izvajanje GWT projekta. Zraven pa dodatno kreira še Ant skripto za gradnjo aplikacije ter konfiguracijske datoteke, ki so potrebne za Eclipse, v kolikor bi se kasneje odločili za uporabo naprednejšega razvojnega okolja in s tem lažjega načina razvoja aplikacije ter razhroščevanja v gostujočem načinu.

Uporaba skripte za kreiranje projekta je preprosta:

```
webAppCreator -out MyApplication com.mycompany.MyApplication
```

Razvoj nove GWT aplikacije s pomočjo Eclipse vtičnika je še preprostejše:

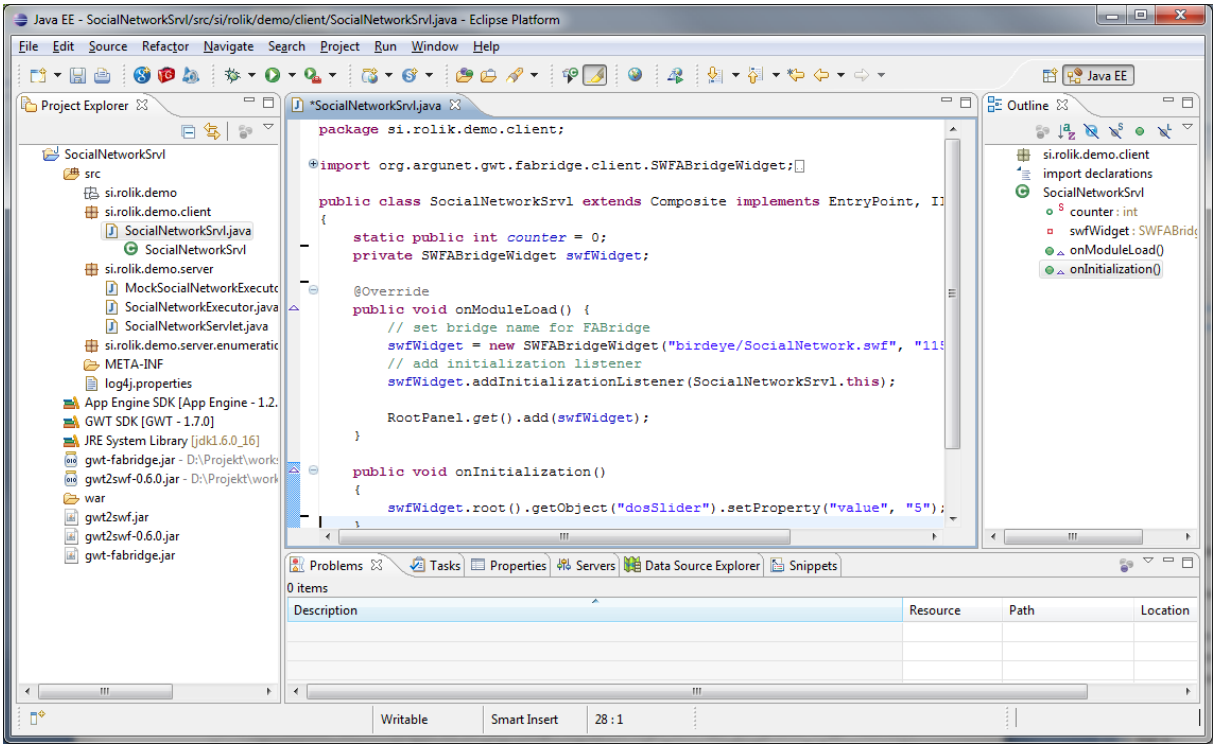
1. V orodni vrstici kliknemo na gumb *New Web Application* Project.
2. Izpolnimo manjkajoče podatke o projektu:
   a. Vnesemo ime projekta.
   b. Vnesemo ime paketa.
   c. Prepričamo se, da sta obkljukani izbiri *Use Google Web Toolkit* ter *use default SDK (App Engine).*
   d. Če nismo namestili SDK, ko smo nameščali Google Plugin for Eclipse vtičnika, je potrebno klikniti na *Configure SDKs*, da določimo, kje se nahaja GWT in *App Engine SDK.*
3. Kliknemo na gumb *Finish.*

4.1.3. Struktura GWT projekta in njegove datoteke

GWT projekt mora ustrezati že vnaprej definirani strukturi, v kolikor želimo, da bo ustrezal pogojem GWT prevajalnika. Projekt mora biti razdeljen na tri podpakete:

- ***Client***: Vsebuje Java kodo s poslovno logiko, ki se bo izvajala na odjemalčevi strani. Ta del kode se prevede v JavaScript. Uporaba Jave tej strani je nekoliko omejena, ker v JavaScript objektno usmerjen koncept ni v celoti implementiran.
- ***Server***: Vsebuje Java kodo s poslovno logiko, namenjeno izvajanju na strežniškem delu aplikacije. V tem delu nimamo omejitev in lahko Javo izkoristimo v polni meri.
- ***WAR*** (oz. *Public* pred verzijo GWT 1.6): Vsebuje datoteke kot so HTML strani, CSS, slike in ikone, ki se v aplikaciji uporabljajo.

Med številnimi datotekami, ki nastanejo ob kreiranju novega projekta, velja omeniti datoteko tipa *gwt.xml*. V njej definiramo, katere dodatne GWT module, javanski razred, kjer se aplikacija začne, ter katera stilna predloga CSS naj se uporabi za oblikovanje spletne aplikacije.



Slika 9: Eclipse razvozno orodje z nameščenim GWT vtičnikom

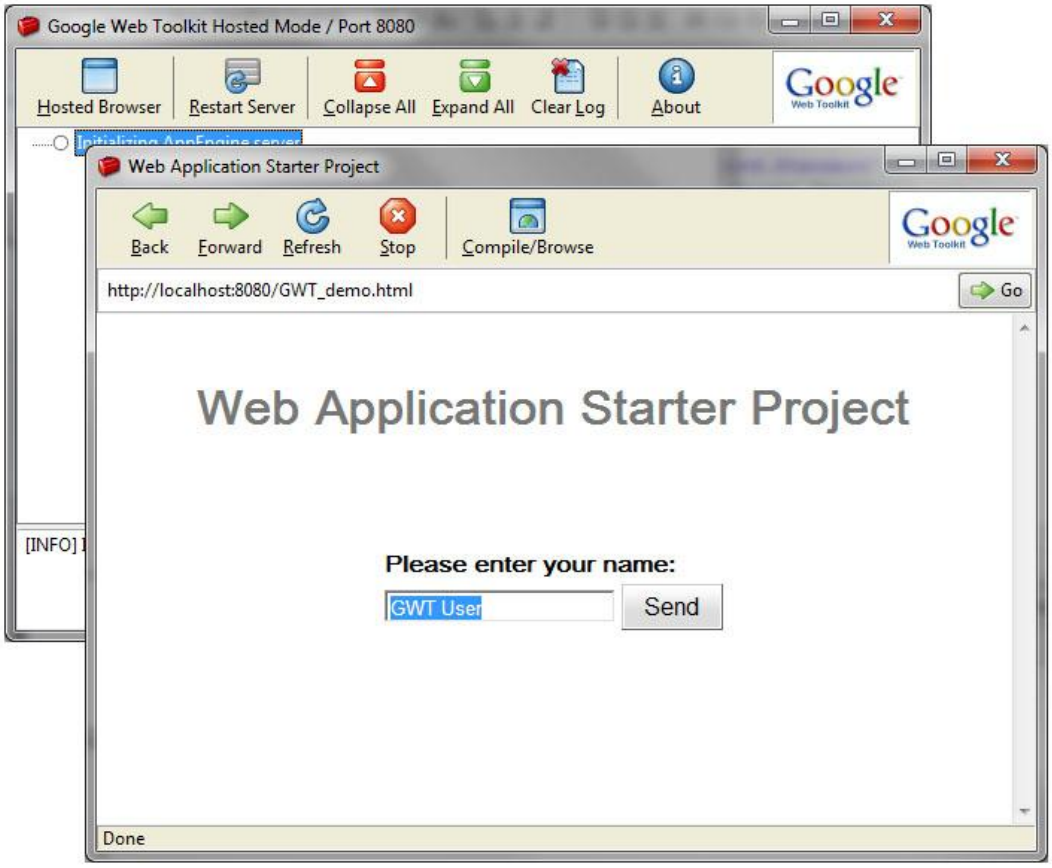
4.1.4. Razvijanje v gostujočem načinu ter namestitev na strežnik

Ko poženemo GWT aplikacijo, se le-ta prikaže v brskalniku podobnem oknu, ki mu Google pravi gostujoči način. Gostujoči način se uporablja samo med razvojem aplikacije. V produkciji bomo aplikacijo poganjali v spletnem načinu. Ta dva načina pa moramo strogo ločiti, saj je med njima kar precejšnja razlika.

Gostujoči način si lahko zamislimo kot pomožna kolesca naše GWT aplikacije. Gre za hibridni razvoj unikatnega okolja v GWT, ki dopušča, da se naša koda izvaja kot prava Javanska koda, vendar še vedno znotraj brskalnika. Izvajanje v gostujočem načinu nadzoruje GWT razvojna lupina.

Razvojna lupina je v bistvu Eclipse Rich Client aplikacija, ki vsebuje konzolo lupine (shell), Tomcat strežnik in enega ali več gostujočih brskalnikov.

Gostujoči brskalnik z razvojno lupino komunicira na dva načina. Prvi je zgolj navadna http povezava, ki prejema spletne strani, CSS datoteke, slike ter ostale vire. Vsi so obdelani preko integriranega Tomcat strežnika z uporabo servleta *com.google.gwt.dev.shell.GWTShellServlet*.



Slika 10: GWT lupina in gostujoči brskalnik

Pri drugem pa gre za nekakšna stranska vrata, ki prestrezajo vse interakcije znotraj gostujočega brskalnika in jih preusmerijo ne k JavaScript, ampak k Java kodi znotraj lupine. Ta Java koda po vrsti kliče našo pravo odjemalčevo Java kodo, ki se je razvojnem okolju prevedla v prenosljivo obliko, znano kot bytecode.

Ker se izvaja Javanska koda, lahko uporabljamo tudi Javanska orodja kot npr. Eclipse razhroščevalnik, findbugs, pmd, JUnit itn.

Ko je koda preverjena, delujoča in stestirana, je naslednji korak prevod kode v obliko, ki se lahko izvaja znotraj navadnih spletnih brskalnikov. Tu pa pride na vrsto spletni način oz. namestitev na strežnik.

Ko v gostujočem brskalniku kliknemo na gumb Compile/Browse, GWT prevajalnik prevede naš odjemalčev (client) paket v JavaScript in pri tem odpre standardni spletni brskalnik za prikaz naše spletne aplikacije. Lokalno ustvarjene datoteke lahko sedaj postavimo na oddaljeni spletni strežnik ter s tem omogočimo javni dostop do naše delujoče spletne aplikacije.

S tako delujočim ogrodjem lahko preidemo na drugi korak. Korak izbire ustrezne komponente za prikaz socialnega omrežja. Vendar pa je bolje, če pred tem obrazložimo pojem socialnega omrežja.

4.2. Pojem socialnega omrežja

Socialno omrežje je ime za skupnost oz. neko skupino posameznikov, imenovanih vozlišča, ki jih vežejo skupni interesi, vizije, norme, pravila, izmenjava denarja/premoženjska korist, trgovina in ostali odnosi in razmerja, ki jih poznamo. [**4**] Socialno omrežje je le skupno ime za raznolike povezave med ljudmi. Motivi za povezovanje med ljudmi so zelo različni.

Poleg socialnih mrež poznamo še [**5**] :

- **Informacijske mreže**: Tipičen primer te vrste so mreže citiranj med znanstvenimi članki. Vozlišča predstavljajo različne članke, med katerimi so usmerjene povezave, v kolikor prvi citira drugega. Ker lahko članki citirajo zgolj pretekle članke, gre v tem primeru očitno za aciklično mrežo, kjer njena topologija oziroma struktura nazorno predstavlja informacijo, shranjeno v vozliščih (člankih). Primera te vrste sta tudi mreža spletnih hiperstrani ter mreža posameznikov v omrežju vsak z vsakim (peer to peer network).
- **Tehnološke mreže**: Vse mreže, ki predstavljajo umetno narejena omrežja, predvsem za oskrbo z določeno dobrino. To so na primer električno ali telefonsko omrežje, internet (fizične povezave), cestno, železniško in letalsko omrežje. Omenimo še, da ima pri takih mrežah pomembno vlogo tudi sama geografska lokacija vozlišč oziroma ustreznih entitet, kar za ostale vrste večinoma ne velja.
- **Biološke mreže**: Mreže so zelo uporabne tudi v biologiji, predvsem na področjih molekularne biologije, genetike ter nevrologije. Zanimiv primer je tudi prehrambena mreža (veriga), sestavljena iz različnih živih bitij, ki so med seboj povezana,v kolikor ustrezajo relaciji plenilec – plen.

Socialne mreže oz. omrežja so sprva raziskovali sociologi in antropologi. Raziskave v mnogih akademskih sferah so pokazale, da socialne mreže lahko ločimo tudi po neke vrste dimenzijah (od družinskih pa do državnih), igrajo pa predvsem pomembno vlogo pri določanju:

- načinov reševanja problemov,
- na kakšne načine so vodene korporacije,
- stopnje uspešnosti posameznikov pri doseganju posameznih ciljev.

V bistvu gre pri socialnih mrežah za različne človeške organizacije in razmerja. Imajo velik pomen pri proučevanju ter analizi skupin, skupnosti ter odnosov med ljudmi. Pomemben je tudi vpliv na posameznike, ki so del tega omrežja.

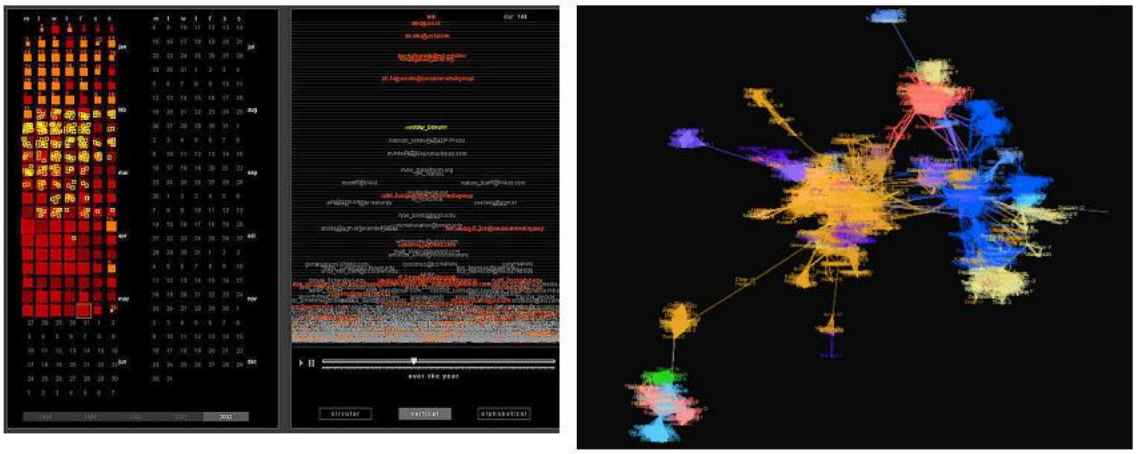
4.3. Izbira primerne komponente za izris socialnega omrežja

4.3.1. Pomen vizualizacije socialnega omrežja

Kot sem že v uvodu omenil, je zahteva podjetja izdelati modul za pregled ter raziskovanje socialnega omrežja. Sama vizualizacija igra pomembno vlogo pri analizi in raziskovanju socialnega omrežja. Vizualizacija je več kot le skopo prikazovanje zanimivih podob, tu gre predvsem za generiranje učljivih situacij ter odkrivanje pomembnih informacij. Podobe znotraj socialnega omrežja ponujajo analitikom nov vpogled v samo strukturo omrežja in jim pomagajo pri podajanju teh vpogledov drugim. V osnovi poznamo dva načina prikazovanja omrežja [**6**]:

- Izrisovanje grafov vozlišč ter povezav med njimi.
- Izpisovanje tabel, kjer stolpci in vrstice prikazujejo ljudi, številke pa povezave med njimi.

V praksi pa se najpogosteje uporablja prikaz s pomočjo grafov.



Slika 11: Slika na levi prikazuje uporabo vizualizacije s pomočjo tabel, na desni pa je primer izrisovanja vozlišč in povezav kompleksnega omrežja

Z vidika preiskovanja naj bi modul omogočal analizo socialnih mrež s pomočjo orodij izločanja in uvrščanja entitet v graf ter bogatenje grafa z različnimi dodatnimi podatki.

4.3.2. Pridobivanje podatkov za prikaz grafov socialnega omrežja

Podatke, ki jih uporabljamo za izrisovanje grafov socialnega omrežja, pridobimo iz policijskih zapisnikov o prometnih nesrečah. Zapisniki so polstrukturirana besedila, ki vsebujejo osnovne podatke o udeležencih nesreče, vozilih ter o nesreči sami. V večini primerov so znana tudi imena policistov, ki so nesreče obravnavali, redko pa tudi

morebitne priče. V nadaljevanju podamo podrobnejši opis podatkov, ki jih poznamo za posamezno entiteto:

- **Policisti**: Znana so zgolj imena policistov.
- **Udeleženci**: Poleg imena poznamo tudi spol osebe, rojstni datum, stalni naslov in državljanstvo. Seveda je poznana tudi vloga udeleženca v prometni nesreči. Lahko gre za voznika ali sovoznika oz. sopotnika ter povzročitelja ali oškodovanca.
- **Vozila**: Poznamo registrsko številko, znamko ter model vozila. Običajno sta znana tudi zavarovalnica, pri kateri je vozilo zavarovano ter lastnik vozila.
- **Dogodek** oz. nesreča: Podan je čas in kraj nesreče, opis poteka ter nestrokovna ocena o vrednosti gmotne škode na vozilih. Pogosto zapisnik vsebuje tudi opis poškodb, ki so jih udeleženci utrpeli.

Določeni podatki za naše namene niso tako pomembni, predvsem gre tu za nekonsistentnost ter nepopolnost. Tako na primer kraj nesreče navadno ni enolično podan, saj vsak policist lokacijo nesreče opredeli na svoj način. Podobno velja za oceno gmotne škode na vozilih ter morda tudi opise poškodb. Tudi znamka in model vozila sta večinoma manj uporabna, saj bi nas zanimal kvečjemu cenovni razred vozila ali pa število sedežev, česar pa iz teh podatkov ne znamo določiti. Na drugi strani pa se izkaže, da določeni podatki ne nosijo pomembnih informacij.

Za namene odkrivanja goljufij je tako smiselno uporabiti zgolj določene podatke oz. atribute. Nekateri izmed njih so statične lastnosti entitet (na primer spol osebe), drugi pa predstavljajo relacije med entitetami – relacijske lastnosti (na primer relacija med voznikom in nesrečo v kateri je bil udeležen).

Pri izrisu socialnega omrežja lahko relacijske atribute predstavimo kot povezave med ustreznimi entitetami. Tako dobimo mreže nesreč, kjer vozlišča predstavljajo same entitete, povezave pa različne relacije med njimi. Na tak način dobimo veliko število različnih mrež, ki se spreminjajo glede na entitete, ki jih vključimo v mreže, ter način, kako jih med seboj povežemo.

Sam izris socialnega omrežja pa je dodatno opredeljen še z uporabniškim vmesnikom, ki naj bi analitikom, glede na zahteve podjetja, omogočal dodatne prilagoditve. Tu gre za spremembe časovnega razpona dogodkov ter možnost izbire policijskih zapisnikov. Filtrira se lahko vsebino (prikaz prič, policistov), ključna pa je izbira oddaljenosti povezanih oseb (koliko vozlišč je med njima).

4.3.3. Izbira primerne rešitve vizualizacije

Implementacija komponente vizualizacije socialnega omrežja iz nič je za podjetje zelo nehvaležna, saj zahteva čas in s tem denar. Zato je bila sprejeta odločitev, da se na spletu poišče ustrezno rešitev med odprtokodnimi projekti za vizualizacijo grafov in omrežij.

Čeprav je na spletu uporaba socialnih mrež zadnje čase zelo razširjena, pa je orodij za vizualizacijo le teh precej manj. Vsi poznamo razne spletne aplikacije, kot so MySpace, Facebook, LinkedIn itd. Njihov namen ni analiza in raziskovanje, ampak

gre tu predvsem za namen socializacije in medspletnega druženja, zato ne ponujajo vizualizacijskih orodij.

Med rešitvami, ki na spletu obstajajo, je precej takšnih, ki so komercialne narave ali pa ne ustrezajo željam in zahtevam, ki smo si jih zadali. Med ustreznimi je zaradi odprtokodnosti in funkcionalnosti najbolj izstopal projekt Birdeye.

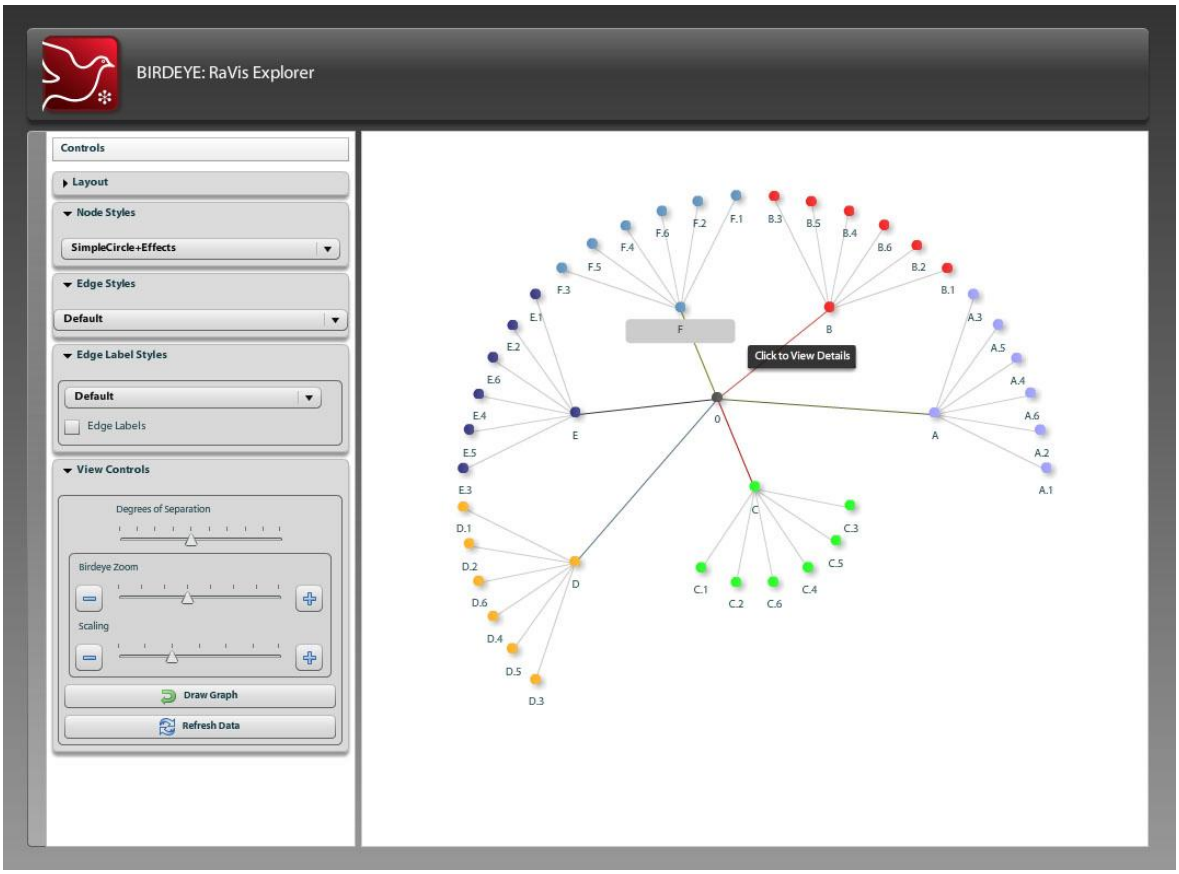
4.3.4. Ravis Birdeye

BirdEye je projekt, ki izstopa po zmogljivosti vizualnih predstavitev, hkrati pa prednjači v razvoju na področju odprtokodnih knjižnic za vizualizacijo podatkov in vizualno analizo v okolju Adobe Flex. [**7**] Gre za zelo zanimivo odprtokodno AS3 (ActionScript 3.0) knjižnico, ki pa žal ne premore uradne dokumentacije.

Znotraj BirdEye prjoketa je vključena tudi komponenta RaVis, ki omogoča analizo sorodnosti, metode za analizo povezav ter povezovanje med objekti.

RaVis je bil razvit iz knjižnice *flexvizgraphlib* osnovane na *Spring Graph* algoritmu za izrisovanje grafov, katerega koda je bila pozneje integrirana v BirdEye projekt in je v času pisanja te diplomske naloge po našem mnenju najbolj napredna komponenta.

Največja pomanjkljivost komponente RaVis oz. celotnega projekta Birdeye je to, da uradna dokumentacija ne obstaja. Na voljo imamo le nekaj primerov uporabe znotraj njihovega SVN repozitorija, Vendar na srečo obstaja spletni forum projekta, ki nam je v pomoč pri manjših težavah.



Slika 12: Birdeye RaVis Explorer – primer uporabe RaVis komponente

RaVis je močna in zelo domišljena komponenta za izris grafov ali omrežij, vendar sem pri delu z njo našel nekaj pomanjkljivosti:

- Knjižnica kot glavni podatkovni vir uporablja XML datoteke. Sicer je mogoče uporabiti tudi druge podatkovne vire, vendar to zaradi pomanjkanje dokumentacije ni tako preprost korak.

- Sama koda, kjer gradimo vizualno obliko komponente, deluje precej zmedeno in neurejeno in je potrebno kar nekaj časa, da se razvijalec znajde v njej. Res je, da gre pri projektu za precej kompleksno zadevo, vendar bi z nekoliko truda lahko stvari zastavili malce bolj modularno.

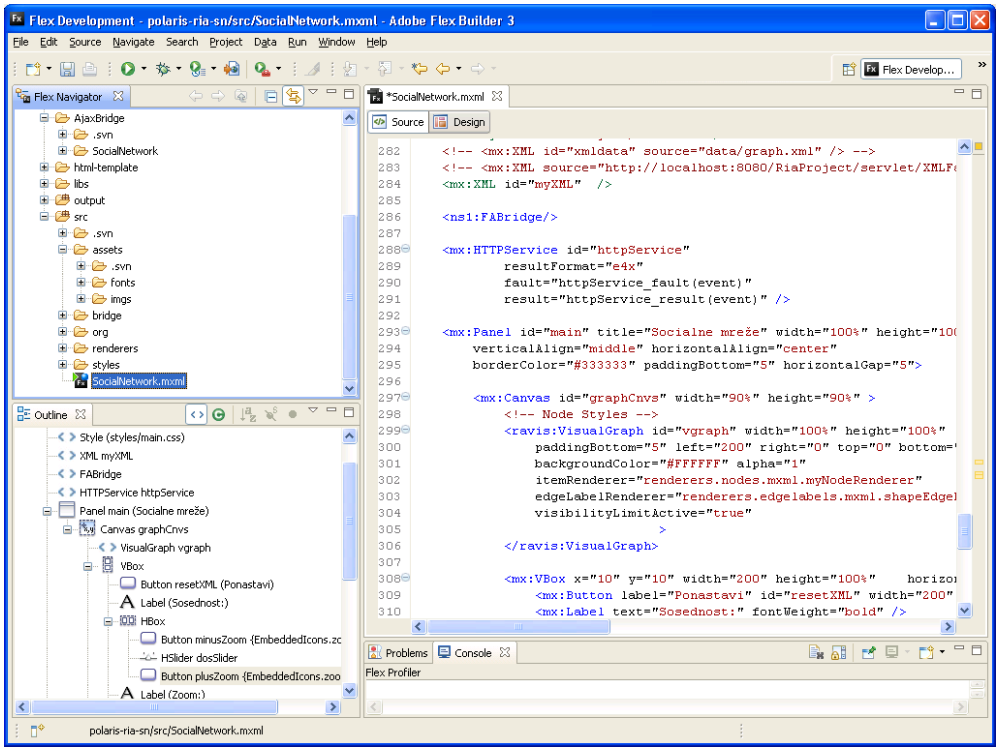
- Kot sem že omenil, projekt nima uradne dokumentacije.

4.3.5. Primer uporabe komponente RaVis za preprost izris omrežja

Najprej seveda potrebujemo graf, oz. podatke, ki jih bomo z grafom prikazali. V našem primeru gre za XML datoteko, katere strukturo bom opisal pozneje. Metoda, ki graf prebere, je poimenovana VisualGraph.

RaVis komponento sestavljajo še metode in objekti kot so:

- item renderer : ki predstavlja vozlišče,
- edge renderer : predstavlja povezavo,
- edge label renderer : predstavlja naziv povezava (Zakaj je ločen od metode edge renderer je nejasno.),
- graph : ta objekt drži podatke o grafu, zbirko vozlišč in povezav brez informacij o dejanskem nalčinu izrisa grafa,
- layouter : ta poskrbi za dejanski prikaz omrežja. Informacije glede načina vizualizacije omrežja se nahajaj znotraj LayoutDrawing objekta.



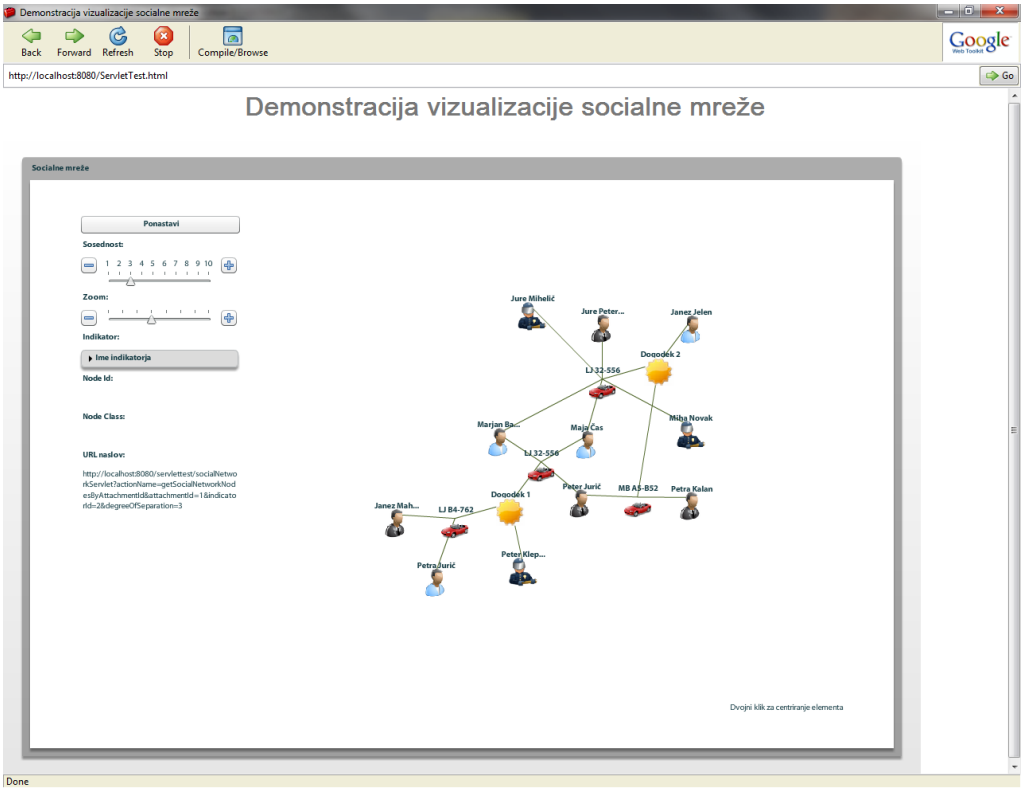
Slika 13: Flex razvojno orodje

Čeprav je za naprednejše načine izrisovanja omrežja potrebno boljše poznavanje objektov in metod komponente RaVis, lahko preprost izris želenega omrežja dosežemo z nekaj preprostimi koraki:

- Poskrbimo za graf s podatki, ki jih želimo prikazati: V našem primeru gre za XML datoteko s podatki o vozliščih in povezavah.
- Sestavimo objekte okrog VisualGrpah metode in jih pravilno nastavimo: Tu gre predvsem za izbiro želenih gradnikov, ki jih bomo uporabili pri sestavi uporabniškega vmesnika.
- Izberemo izvorno vozlišče (root node): Čeprav se sliši malenkostno, gre za pomemben korak, saj se graf brez izbranega izvornega vozlišča ne prikaže pravilno. To storimo znotraj metode InitData():

```
startRoot = graph.nodeByStringId("1").vnode;
```

- Zadnji korak je Izris omrežja.



Slika 14: Primer izrisa testnega socialnega omrežja, ki se za namen razvoja uporablja v podjetju Optilab d.o.o.

4.3.6. XML datoteka kot podatkovni vir RaVis komponente

Spodnji primer XML datoteke prikazuje način zapisa podatkov, ki jih potrebujemo za izris omrežja.

Primer XML datoteke:

```
<Graph>
<Node id="1" name="Vozlišče A" desc="Opis vozlišča A" nodeColor="0x333333" nodeSize="32"
nodeClass="event" nodeIcon="assets/imgs/event.png" x="10" y="10" />
<Node id="1" name="Vozlišče B" desc="Opis vozlišča B" nodeColor="0x333333" nodeSize="32"
nodeClass="event" nodeIcon="assets/imgs/event.png" x="10" y="10" />
<Edge fromID="1" toID="2" edgeLabel="Naziv povezave" flow="50" color="0x556b2f"
edgeClass="person" edgeIcon="NoChange" />
</Graph>
```

Gre za vnaprej definirano XML arhitekturo s preprosto enonivojsko shemo, ki jo sestavljata le dva elementa:

- **Vozlišče** (Node): Vsako vozlišče mora biti definirano z unikatnim identifikatorjem. Znotraj vozlišča pa lahko dodatno definiramo še Ime vozlišča, opis, barvo, velikost (če gre za način vektorske vizualizacije), definicijski razred, ikono ter pozicijske koordinate.
- **Povezava** (Edge): Povezava mora biti definirana z uporabo dveh identifikatorjev, ki predstavljata različni vozlišči. Dodatno lahko povezavi dodelimo še ime, debelino, barvo, definicijski razred ter ikono.

Možnosti ustvarjanja različnih vizualnih predstavitev podatkov, s pomočjo po želji oblikovanih elementov znotraj XML, so tako neomejene.

S tem lahko zaključimo izbiro ustrezne komponente in izvedemo še zadnji korak, korak vzpostavitve komunikacije med različnima tehnologijama.

4.4. Komunikacija med SWF in GWT

Izbira Flex komponente nas je privedla do novega izziva. Izbrano komponento je potrebno integrirati v GWT ogrodje ter z njo vzpostaviti ustrezno komunikacijo.

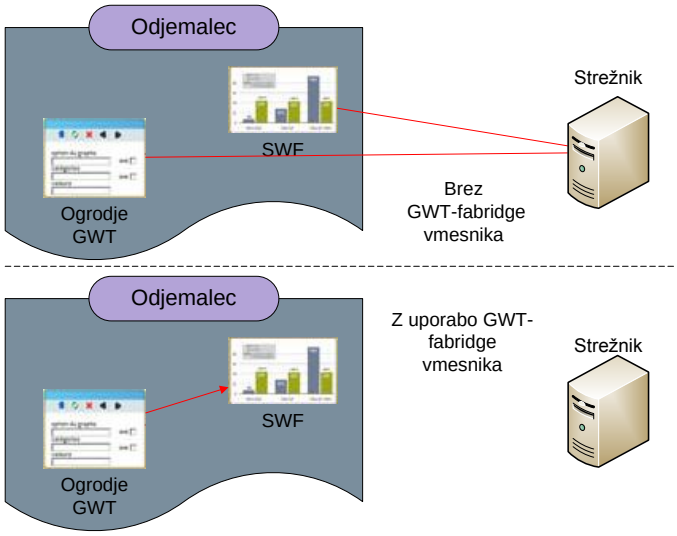
Modul za prikaz socialnih mrež je interaktivne narave. Kar pomeni, da lahko uporabnik z gumbi in ostalimi kontrolniki po želji oblikuje način izrisa socialnega omrežja. Vsak uporabnikov klik znotraj komponente, se ob potrditvi prek ogrodja pošlje na strežniško stran. Strežnik prebere zahtevo uporabnika, nato pa glede na zahtevo pošlje odjemalcu podatke, ki so ustrezno zapisani v XML obliki.

Potrebno je torej vzpostaviti način medsebojne komunikacije med GWT ogrodjem ter Flex komponento, ki je, kot Flex produkt, tipa SWF. SWF je v osnovi namenjen predvajanju multimedijske vsebine, vendar Flex spomočjo dodatnih vmesnikov omogoča dodatni nadzor in kontrolo nad vsebino izvajanja. Kot sem omenil že v poglavju z opisom FLEX tehnologije, Adobe priporoča vmesnik FABridge kot najprimernejši za vzpostavitev te vrste komunikacije med Flex-om in AJAX-om.

4.4.1. GWT-Fabridge

Flex AJAX Bridge oz. FABridge je manjša knjižnica, ki je že del Flex SDK paketa in jo lahko vstavimo v Flex aplikacijo, Flex komponento ali pa SWF datoteko in tako do nje preko skriptnega jezika, kot npr. JavaScript, omogočimo dostop iz brskalnika.

GWT-FABridge je GWT vmesnik, omogoča uporabo FABridge v GWT aplikaciji ter preko nadgradnje gwt2swf vmesnika omogoča upravljanje SWF datotek. Če poenostavimo, je GWT-Fabride vmesnik za vzpostavitev komunikacije med Flex-om in GWT. [8]



Slika 15: Način nadzora SWF datotek z ali brez GWT-fabridge vmesnika

Z uporabo vmesnikov GWT-FABridge in gwt2swf je upravljanje SWF datotek dokaj preprosto:

Najprej poskrbimo, da ta dva nova modula vključimo v našo GWT aplikacijo. To storimo z vpisom v gwt.xml datoteko:

```
<!-- Other module inherits                                -->
<inherits name='pl.rmalinowski.gwt2swf.GWT2SWF' />
<inherits name='org.argunet.gwt.fabridge.FABridge' />
```

Nato pa v razredu, kjer imamo namen prikazati graf, kreiramo nov SWFabridge gradnik, ki ga bomo uporabili za branje in prikaz SWF datoteke:

```
// set bridge name for FABridge
swfWidget = new SWFABridgeWidget("app.swf", 400, 400, "example");
// add swfWidget to RootPanel
RootPanel.get().add(swfWidget);
// show swf media
swfWidget.setVisible(true);
```

Kot primer komunikacije s strani GWT do SWF lahko uporabimo nastavljanje potrditvenega polja (checkbox):

```
// changes checkbox status in swfWidget
    setCheckBox.addClickHandler(new ClickHandler()
    {
      public void onClick(ClickEvent arg0)
      {
        // get check box
        BridgeObject check = swfWidget.root().getObject("check");
        // get check box status
        Boolean status = new Boolean(check.getProperty("selected").getContent().toString());
        // change check box status
        check.setProperty("selected", !status.booleanValue());
      }
```

```
        });
```

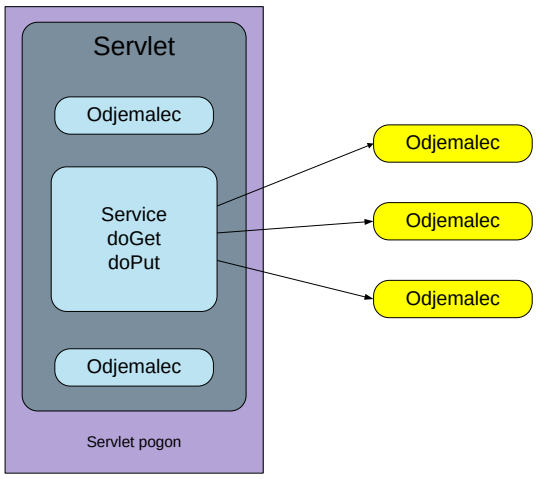
Za primer komunikacije strani SWF do GWT pa vzemimo recimo premikanje drsnika znotraj SWF:

```
// add change listener to flex slider
swfWidget.root().getObject("slider").addEventListener("change", new ISWFListener() {
        public void onSWFEvent(JavaScriptObject event) {
                // wrap event object into bridge object
                BridgeObject e = new BridgeObject(event);
                // get slider value from event object
                sliderLabel.setText(e.getProperty("value").getContent().toString());
                }
        });
}
```

GWT-FABridge je izdredno uporaben GWT vmesnik z dokaj preprostim načinom uporabe. Vendar pa je v primeru takšnega načina komunikacije potrebno dodatno programirati Flex knjižnice. Tu pa, v našem primeru, naletimo na nekaj težav, saj je, kot sem že omenil, koda znotraj RaVis komponente precej neurejena.

4.4.2. Uporaba servlet načina podajanja podatkov

Rešitev, ki bi zmanjšala dodatno programiranje Flex komponente, je bila ideja o storitvi, ki bi RaVis komponenti nudila direktni dostop do XML podatkovnega vira. Najenostavnejša rešitev za ta problem je bila uporaba preprostega Java servleta, ki je opravila zgoraj omenjeno nalogo. Čeprav sem uporabil standardni Java servlet, znotraj GWT ni bilo težav, saj se servlet izvaja na strežniškem delu in se ne prevaja v JavaScript.



Slika 16: Prikazuje razmerje med več zahtevami odjemalca ter eno servlet instanco.

Servlet je objekt, katerega osnovni namen je, da sprejema zahtevke (request) in zanje generira odgovor (response). Definiran je s paketom *javax.servlet.http,* ki združuje HTTP podrazrede generičnih elementov vključno z objekti za vzdrževanje seje, ki sledijo zahtevkom in odgovorom med strežnikom in odjemalcem. Servlete lahko vključimo paket znotraj WAR datoteke ali znotraj spletnih aplikacij.

Življenski krog servleta je sestavljen iz naslednjih korakov:

- Servlet razred se ob pomoči vsebnika (container) naloži ob zagonu aplikacije.
- Vsebnik (container) pokliče *init()* metodo. Ta metoda inicializira servlet in mora biti klicana preden servlet začne obdelovati zahteve (request). Skozi celotni obstoj servleta se *init()* metoda pokliče le enkrat.
- Po inicilizaciji servlet lahko prične obdelovati strežniške zahtevke. Vsak zahtevek je obdelan znotraj posameznih nizov. Vsebnik pokliče *service()* metodo za vsak zahtevek. *Service()* metoda določi vrsto zahtevka in poskrbi za pravo metodo, ki bo zahtevek oskrbela. Razvijalec mora v servletu poskrbeti za pravilno implementacijo teh metod. Če zahtevek nima ustrezno prirejene metode, bo nadrejena metoda vrnila napako za določeni zahtevek.
- Na koncu, vsebnik pokliče metodo *destroy()*, ki zaključi delovanje servleta. *Destroy()* metoda se kot *init()* pokliče le enkrat v življenskem krogu servleta.

Servlet se je izkazal kot zelo uporaben način pri reševanju težav podajanja podatkov. Saj sem s preprostim programiranjem *doGet()* metode in nekaj dodatnega programiranja v MXML dosegel dokaj preprost in učinkovit način branja zahtevanega XML zapisa.

5. Sklepne ugotovitve

Ugotovimo lahko, da je RIA postala zrela tehnologija, ki postaja redna praksa razvijalcev in je že skoraj nadomestila namizne aplikacije. Z uporabo primerne tehnologije za razvoj obogatenih aplikacij lahko pomembno vplivamo na strateško načrtovanje ter hitrost in ceno razvoja pri izgradnji aplikacij. Žal pri izbiri RIA tehnologij nisem imel dovolj časa, da bi raziskal in preizkusil vsa orodja, ki so v spletu na voljo. Zato je bila glavna utež izbiranja postavljena na krivuljo učenja ter čas prilagajanja novemu orodju.

GWT se je izkazal kot zelo primerno ogrodje. Arhitekturo GWT projektov in Java programski jezik omogoča izkušenim Java programerjem, da se brez težav prilagodijo načinu razvijanja obogatenih spletnih aplikacij. Strateško gledano se za prihodnost projekta GWT ni bati. Projekt ima dobro finančno in razvojno zaledje podjetja Google in je v stalnem razvoju in nadgrajevanju. V letošnjem letu je Google izdal dve novi verziji GWT orodja. V času pisanja te diplomske naloge sem bil sam priča spremembi verzije iz 1.6 v 1.7.

Uveljavljeni Java programski jezik je dobro dokumentiran in integracija GWT ogrodja z drugimi tehnologijami je zaradi uporabe Java programskega jezika preprostejša. V diplomski nalogi mi je uspelo pokazati kako preprosto je, z uporabo nekaj vmesnikov oz. knjižnic, povezati dve komponenti različnih tehnologij. Kompleksnost posamezne komponente pri samem povezovanju ne igra pomembne vloge.

Izbiro komponente za vizualizacijo socialnega omrežja, lahko vzamemo kot primer kompleksne tehnologije. Modul za prikaz socialnega omrežja še ni povsem dokončan, saj gre za kompleksnejši primer, kjer bo potrebno s pomočjo izkopavanja podatkov še dopolniti izris in iskanje sumljivih vzorcev znotraj socialnega omrežja. Prav tako je v razvoju tudi sam uporabniški vmesnik. Le-ta bo namreč še dopolnjen z dodatnimi zahtevami in željami kasnejših uporabnikov.

6. Priloge

6.1. Seznam slik

6.2. Literatura

[1] "Rich Internet application." Dostopno na: http://en.wikipedia.org/wiki/Rich_Internet_application

[2] "Google Web Toolkit." Dostopno na: http://en.wikipedia.org/wiki/Google_Web_Toolkit

[3] "Open Laszlo." Dostopno na: http://en.wikipedia.org/wiki/Open_Laszlo

[4] "Social network." Dostopno na: http://en.wikipedia.org/wiki/Social_network

[5] Newman, M.: "The structure and function of complex networks.", 167–256 (2003)

[6] Fernanda B. Viégas, Judith Donath: "Social Network Visualization: Can We Go Beyond the Graph?" Workshop on Social Networks 4, str. 6-10 (2004)

[7] Jason Bellone and Daniel Lang: "Visualizing Relational Data Using Graph Theory." Flash&Flex, str. 24-27 (2008)

[8] "GWT-fabridge official blog." Dostopno na: http://1pxsolidblack.blogspot.com/

[9] Yakov Fain, Dr. Victor Rasputnis, Anatole Tartakovsk:"Rich Internet Applications with Adobe Flex & Java (Secrets of the Masters)." (2007)

[10] Hinchcliffe, D.: "Seven things every software project needs to know about Ajax." (2006) Dostopno na: http://www.linkegypt.com/blogs/b/Seven-Things-Every-Software-Project-Needs-to-Know-About-Ajax/38/Seven-Things-Every-Software-Project-Needs-to-Know-About-Ajax.html

[11] Dave Crane, Eric Pascarello, Darren James:"Ajax in Action." (2005)

[12] "Canoo - Ultra Light Client." Dostopno na: http://www.canoo.com/ulc/home/whatisulc.html

[13] "Backbase." Dostopno na: http://www.backbase.com/products/rich-portal/#technology_overview

[14] "How Nexaweb works." Dostopno na: http://www.nexaweb.com/home/us/index.html@cid=2295.html

[15] Robert Cooper, Charles Collins:"GWT in Practice." (2008)